3/3/2 (Item 2 from file: 351)
DIALOG(R)File 351:Derwent WPI

014190748 **Image available**
WPI Acc No: 2002-011445/200201
XRPX Acc No: N02-009461

Job assigning method for parallel processing method includes parallel computer

Patent Assignee: TAISHO PHARM CO LTD (TAIS); HONDA MOTOR CO LTD (HOND); FUJI XEROX CO LTD (XERF); INABATA S (INAB-I); KITAMURA K (KITA-I); MIYAKAWA N (MIYA-I); NAGASHIMA U (NAGA-I); TAKASHIMA H (TAKA-I); YAMADA S (YAMA-I)

Inventor: INABATA S; KITAMURA K; MIYAKAWA N; NAGASHIMA U; TAKASHIMA H; YAMADA S

Number of Countries: 096 Number of Patents: 008

Patent Family:

Patent No	Kind	Date	Applicat No	Kind	Date	Week	
WO 200184343	A1	20011108	WO 2001JP3554	A	20010425	200201	B
JP 2001312485	A	20011109	JP 2000131253	A	20000428	200207	
AU 200152561	A	20011112	AU 200152561	A	20010425	200222	
EP 1286273	A1	20030226	EP 2001925891	A	20010425	200319	
			WO 2001JP3554	A	20010425		
US 20030204576	A1	20031030	WO 2001JP3554	A	20010425	200372	
			US 2002257913	A	20021018		
CN 1439130	A	20030827	CN 2001811717	A	20010425	200375	
AU 771360	B2	20040318	AU 200152561	A	20010425	200454	
CN 1655135	A	20050817	CN 2001811717	A	20010425	200572	
			CN 20059542	A	20010425		

Priority Applications (No Type Date): JP 2000131253 A 20000428

Patent Details:

Patent No	Kind	Lan	Pg	Main IPC	Filing Notes
WO 200184343	A1	J	98	G06F-015/177	
Designated States (National): AE AG AL AM AT AU AZ BA BB BG BR BY BZ CA CH CN CO CR CU CZ DE DK DM DZ EE ES FI GB GD GE GH GM HR HU ID IL IN IS KE KG KP KR KZ LC LK LR LS LT LU LV MA MD MG MK MN MW MX MZ NO NZ PL PT RO RU SD SE SG SI SK SL TJ TM TR TT TZ UA UG US UZ VN YU ZA ZW					
Designated States (Regional): AT BE CH CY DE DK EA ES FI FR GB GH GM GR IE IT KE LS LU MC MW MZ NL OA PT SD SE SL SZ TR TZ UG ZW					
JP 2001312485	A		30	G06F-015/177	
AU 200152561	A				Based on patent WO 200184343
EP 1286273	A1	E		G06F-015/177	Based on patent WO 200184343
Designated States (Regional): AL AT BE CH CY DE DK ES FI FR GB GR IE IT LI LT LU LV MC MK NL PT RO SE SI TR					
US 20030204576	A1			G06F-015/177	
CN 1439130	A			G06F-015/177	
AU 771360	B2			G06F-015/177	Previous Publ. patent AU 200152561 Based on patent WO 200184343
CN 1655135	A			G06F-015/16	Div ex application CN 2001811717

PATENT ABSTRACTS OF JAPAN

(11)Publication number : 2001-312485
(43)Date of publication of application : 09.11.2001

(51)Int.CI. G06F 15/177
G06F 9/46
G06F 15/16

(21)Application number : 2000-131253
(22)Date of filing : 28.04.2000

(71)Applicant : FUJI XEROX CO LTD
TAISHO PHARMACEUT CO LTD

(72)Inventor : YAMADA SOU
INAHATA SHINJIROU
MIYAGAWA NOBUAKI
TAKASHIMA HAJIME
KITAMURA KAZUYASU
NAGASHIMA UNPEI

(54) ALLOCATING METHOD FOR JOB BY PARALLEL PROCESSING METHOD AND PARALLEL PROCESSING METHOD

(57)Abstract:
PROBLEM TO BE SOLVED: To efficiently allocate jobs to respective processors when parallel processing is performed by using a parallel computer constituted by connecting a host computer and plural processors by a common bus.
SOLUTION: A job whose ratio of a communication time and an operation time is specific or larger than several parts of that of a processor and a job which is smaller are allocated alternately to each processor. Plural processors and plural jobs are divided into plural groups while made to correspond to each other, jobs which are close in the size consisting of the communication time and operation time and jobs which are close in the ratio of the communication time and operation time are put in different job groups, and jobs which are close in the size consisting of the communication time and operation time and the ratio of the communication time and operation time are so allocated that the allocation places of the jobs in the job groups are mutually different among plural job groups.

待ち行列

0(0)
32(1)
1(2)
31(3)
2(4)
⋮
15(30)
17(31)
16(32)

LEGAL STATUS

[Date of request for examination]

[Date of sending the examiner's decision of rejection]

[Kind of final disposal of application other than the examiner's decision of rejection or application converted registration]

[Date of final disposal for application]
[Patent number]
[Date of registration]
[Number of appeal against examiner's decision of rejection]
[Date of requesting appeal against examiner's decision of rejection]
[Date of extinction of right]

(19)日本国特許庁（ＪＰ）　(12)公開特許公報（Ａ）　(11)特許出願公開番号
特開2001－312485
（P2001－312485A）
(43)公開日　平成13年11月9日(2001.11.9)

(51)Int.Cl.[7]	識別記号	FI	テーマコード(参考)
G06F 15/177	674	G06F 15/177	674C 5B045
9/46	360	9/46	360B 5B098
15/16	610	15/16	610B

審査請求　未請求　請求項の数14　ＯＬ　（全 30 頁）

(21)出願番号　特願2000－131253(P2000－131253)

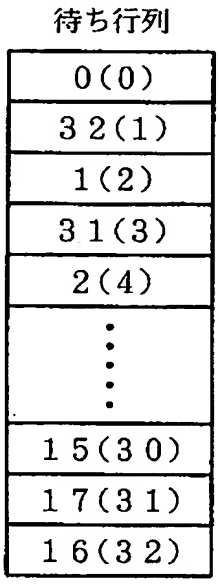
(22)出願日　平成12年４月28日(2000.4.28)

(71)出願人　000005496
富士ゼロックス株式会社
東京都港区赤坂二丁目17番22号
(71)出願人　000002819
大正製薬株式会社
東京都豊島区高田３丁目24番１号
(72)発明者　山田　想
神奈川県足柄上郡中井町境430　グリーンテクなかい　富士ゼロックス株式会社内
(74)代理人　100091546
弁理士　佐藤　正美

最終頁に続く

(54)【発明の名称】　並列処理方法におけるジョブの割り当て方法および並列処理方法

(57)【要約】
【課題】　ホスト計算機と、複数のプロセッサとが共通のバスにより接続されて構成される並列計算機を用いて並列処理を行う場合において、効率良く、各プロセッサにジョブの割り当てを行う。
【解決手段】　各プロセッサには、通信時間と演算時間との比が、所定の値あるいはプロセッサの数分の１よりも大きいジョブと、小さいジョブとを交互に割り当てる。または、複数個のプロセッサと、複数個のジョブとを、対応させて複数個のグループに分割し、通信時間と演算時間とからなるサイズと、前記通信時間と演算時間との比とが近似するジョブは、ジョブ・グループの異なるものにそれぞれ属するようにし、かつ、通信時間と演算時間とからなるサイズと、通信時間と演算時間との比とが近似するジョブの、ジョブ・グループのそれぞれ内での割り当て順番が、複数のジョブ・グループの間において互いに異なるように割り当てる。



1

【特許請求の範囲】

【請求項１】ホスト計算機と、このホスト計算機と共通のバスを通じて接続される複数のプロセッサにより構成される計算機システムを用いて、前記ホスト計算機と前記プロセッサとの間の通信処理と前記プロセッサにおける演算処理とからなる複数個のジョブを、前記複数個のプロセッサにより並列に処理する並列処理方法において、

前記プロセッサのそれぞれには、前記ジョブの通信時間と演算時間との比の平均値が、所定の値になるように各ジョブを割り当てることを特徴とする並列処理方法におけるジョブの割り当て方法。

【請求項２】請求項１に記載の並列処理方法におけるジョブの割り当て方法において、

前記複数個のジョブの全体の通信時間と、前記複数個のジョブを前記複数個のプロセッサに均等に割り当てたとしたときの一つのジョブにおける演算時間の総計とが、ほぼ等しい時には、前記プロセッサのそれぞれには、前記ジョブの通信時間と演算時間との比の平均値が、前記プロセッサの数分の１になるように各ジョブを割り当てることを特徴とする並列処理方法におけるジョブの割り当て方法。

【請求項３】請求項１または請求項２に記載の並列処理方法におけるジョブの割り当て方法において、

各プロセッサには、通信時間と演算時間との比が、前記所定の値あるいは前記プロセッサの数分の１よりも大きいジョブと、小さいジョブとを交互に割り当てることを特徴とする並列処理方法におけるジョブの割り当て方法。

【請求項４】ホスト計算機と、このホスト計算機と共通のバスを通じて接続される複数のプロセッサにより構成される計算機システムを用いて、前記ホスト計算機と前記プロセッサとの間の通信処理と前記プロセッサにおける演算処理とからなるジョブの複数個を、前記複数個のプロセッサにより並列に処理する並列処理方法において、

前記複数個のジョブには、Ｎ（Ｎは整数）からＭ（Ｎより大きい整数）まで連続する整数で番号付けを行い、

直前に割り当ての済んだジョブの番号Ｊが（Ｎ＋Ｍ）／２以下であれば、番号がＮ＋（Ｍ－Ｊ－１）のジョブを次のジョブとして割り当て、

直前に割り当ての済んだジョブの番号Ｊが（Ｎ＋Ｍ）／２より大きいならば、番号がＮ＋（Ｍ－Ｊ）のジョブを次のジョブとして割り当てるまたは、

直前に割り当ての済んだジョブの番号Ｊが（Ｎ＋Ｍ）／２より小さいならば、番号がＮ＋（Ｍ－Ｊ）のジョブを次のジョブとして割り当て、

直前に割り当ての済んだジョブの番号Ｊが（Ｎ＋Ｍ）／２以上であれば、番号がＮ＋（Ｍ－Ｊ－１）のジョブを次のジョブとして割り当てて、

2

前記通信時間と前記演算時間との比の平均値が、所定の値になるように各ジョブを割り当てることを特徴とする並列処理方法におけるジョブの割り当て方法。

【請求項５】ホスト計算機と、このホスト計算機と共通のバスを通じて接続される複数のプロセッサにより構成される計算機システムを用いて、前記ホスト計算機と前記プロセッサとの間の通信処理と前記プロセッサにおける演算処理とからなるジョブの複数個を、前記複数個のプロセッサにより並列に処理する並列処理方法において、

前記複数個のプロセッサを、グループ内の前記プロセッサの数が等しいＧ個（Ｇは２以上の整数）のプロセッサ・グループに分割すると共に、前記複数個のジョブを、グループ内の前記ジョブの数がほぼ等しいＧ個のジョブ・グループに分割し、

前記プロセッサ・グループと、前記ジョブ・グループとを１対１に対応させて、一つのジョブ・グループ内のジョブを、対応する一つのプロセッサ・グループ内のプロセッサに割り当てるものであって、

前記通信時間と前記演算時間とからなるサイズと、前記通信時間と前記演算時間との比とが近似するジョブは、前記ジョブ・グループの異なるものにそれぞれ属するようにすると共に、

前記通信時間と前記演算時間とからなるサイズと、前記通信時間と演算時間との比とが近似するジョブの、前記ジョブ・グループのそれぞれ内での割り当て順番を、前記複数のジョブ・グループにおいて互いに異なるように割り当てることを特徴とする並列処理方法におけるジョブの割り当て方法。

【請求項６】請求項５に記載の並列処理方法におけるジョブの割り当て方法において、

前記複数個のジョブには、Ｎ（Ｎは整数）からＭ（Ｎより大きい整数）まで連続する整数で番号付けを行い、

ｇ番目のジョブ・グループは、前記ジョブの番号が０から順に番号付けされたものに置き換えられたとき、前記Ｇで除算した余りがｇに等しい番号のジョブにより構成されるようにして、

前記通信時間と前記演算時間との比の平均値が、所定の値になるように各ジョブを割り当てることを特徴とする並列処理方法におけるジョブの割り当て方法。

【請求項７】請求項５に記載の並列処理方法におけるジョブの割り当て方法において、

前記複数個のジョブには、Ｎ（Ｎは整数）からＭ（Ｎより大きい整数）まで連続する整数で番号付けを行い、

ｇ番目のジョブ・グループは、前記ジョブの番号が０から順に番号付けされたものに置き換えられたとき、

ジョブ番号を２×Ｇで除算した余りがｇに等しいジョブおよびジョブ番号を２×Ｇで除算した余りが（２×Ｇ－ｇ－１）に等しいジョブにより構成されるようにして、

前記通信時間と前記演算時間との比の平均値が、所定の

値になるように各ジョブを割り当てることを特徴とする並列処理方法におけるジョブの割り当て方法。

【請求項８】請求項６または請求項７に記載の並列処理方法におけるジョブの割り当て方法において、

ｇ番目のジョブ・グループを構成するジョブのうち、最初にプロセッサに割り当てられるジョブは、

ジョブ番号Ｊをプロセッサ・グループ数Ｇで除算した時の商が互いに等しいジョブにより構成され、その商を用いて、０以上、Ｂ＝（Ｍ－Ｎ＋１）／Ｇ以下の整数で番号付けされるジョブ・ブロックのうち、

（Ｂ／Ｇ）×ｇ以上、（Ｂ／Ｇ）×（ｇ＋１）未満の番号を有するジョブ・ブロックに含まれる番号のジョブであることを特徴とする並列処理方法におけるジョブの割り当て方法。

【請求項９】請求項６または請求項７に記載の並列処理方法におけるジョブの割り当て方法において、

対応するプロセッサ・グループを構成するプロセッサ・エレメント数がｍであるｇ番目のジョブ・グループを構成するジョブのうち、最初にプロセッサに割り当てられるジョブは、

ジョブ番号Ｊをプロセッサ・グループ数Ｇで除算した時の商が互いに等しいジョブにより構成され、その商を用いて、０以上、Ｂ＝（Ｍ－Ｎ＋１）／Ｇ以下の整数で番号付けされるジョブ・ブロックのうち、

（Ｂ／Ｇ／ｍ）×ｇ以上、（Ｂ／Ｇ／ｍ）×（ｇ＋１）未満の番号を有するジョブ・ブロックに含まれる番号のジョブであることを特徴とする並列処理方法におけるジョブの割り当て方法。

【請求項１０】請求項８に記載の並列処理方法におけるジョブの割り当て方法において、

ｇ番目のジョブ・グループにおいて直前に割り当ての済んだジョブがｂ番目のジョブ・ブロックに含まれる場合に、

（ｂ＋１）番目のジョブ・ブロックにｇ番目のジョブ・グループのジョブがあれば（ｂ＋１）番目のジョブ・ブロック内のジョブを、そうでなければ０番目のジョブ・ブロック内のジョブを、

または（ｂ－１）が０以上なら（ｂ－１）番目のジョブ・ブロック内のジョブを、そうでなければｇ番目のジョブ・グループ内でまだ割り当ての行われていない最大番号のジョブを、

前記ｇ番目のジョブ・グループの次のジョブとして割り当てることを特徴とする並列処理方法におけるジョブの割り当て方法。

【請求項１１】請求項９に記載の並列処理方法におけるジョブの割り当て方法において、

ｇ番目のジョブ・グループにおいて直前に割り当ての済んだジョブがｂ番目のジョブ・ブロックに含まれる場合に、

ｂ＋（Ｂ／ｍ）番目のジョブ・ブロックにｇ番目のジョブ・グループのジョブがあればｂ＋（Ｂ／ｍ）番目のジョブ・ブロック内のジョブを、そうでなければｂ＋１を（Ｂ／ｍ）で除算した余りに等しい番号のジョブ・ブロック内のジョブを、

またはｂ－（Ｂ／ｍ）が０以上ならｂ－（Ｂ／ｍ）番目のジョブ・ブロック内のジョブを、ｂが０なら（Ｂ／ｍ）で除算した余りが（Ｂ／ｍ）－１に等しいジョブ・ブロック番号のうちｇ番目のジョブ・グループのジョブを含む最大の番号のジョブ・ブロック内のジョブを、ｂがそれ以外であれば、（Ｂ／ｍ）で除算した余りが、（ｂ－１）を（Ｂ／ｍ）で除算した余りと等しいジョブ・ブロック番号のうちｇ番目のジョブ・グループのジョブを含む最大の番号のジョブ・ブロック内のジョブを、前記ｇ番目のジョブ・グループの次のジョブとして割り当てることを特徴とする並列処理方法におけるジョブの割り当て方法。

【請求項１２】ｎ個（ｎは正の整数）の縮約シェルを用いて表現される分子のエネルギーを計算する分子軌道計算を、ホスト計算機と複数個のプロセッサとにより構成される計算機システムを用いて行う並列処理方法であって、

縮約シェルＲ，Ｓ，Ｔ，Ｕのそれぞれに含まれる原始シェルｒ，ｓ，ｔ，ｕのそれぞれの成分である原始基底関数ｉ，ｊ，ｋ，ｌをインデックスとして用いて表わされる２電子積分関数ｇの関数値ｇ（ｉ，ｊ，ｋ，ｌ）と；

前記原始基底関数ｋをひとつの構成要素とする縮約基底関数Ｋおよび前記原始基底関数ｌをひとつの構成要素とする縮約基底関数Ｌとをインデックスとして用いて表わされる密度行列Ｐの要素Ｐ（Ｋ，Ｌ）と；係数Ａ１との積Ａ１・Ｐ（Ｋ，Ｌ）・ｇ（ｉ，ｊ，ｋ，ｌ）の全ての縮約基底関数に関する総和ｆ１（Ｉ，Ｊ）と、

前記２電子積分関数の関数値ｇ（ｉ，ｋ，ｊ，ｌ）と；前記密度行列Ｐの要素Ｐ（Ｋ，Ｌ）と；係数Ａ２との積Ａ２・Ｐ（Ｋ，Ｌ）・ｇ（ｉ，ｋ，ｊ，ｌ）の全ての縮約基底関数に関する総和ｆ２（Ｉ，Ｊ）との和ｆ（Ｉ，Ｊ）＝ｆ１（Ｉ，Ｊ）＋ｆ２（Ｉ，Ｊ）の、前記原始基底関数ｉ，ｊをひとつの構成要素とする前記縮約基底関数Ｉ，Ｊに含まれる全ての前記原始基底関数に関する和で表わされるフォック行列Ｆの全ての行列要素Ｆ（Ｉ，Ｊ）の計算を、

最も外側のループは、Ｒ≦ｎmax（ｎmax はｎ個の縮約シェルに付与された番号の最大値）およびＴ≦Ｒなる関係を満たす前記縮約シェルＲとＴとの組み合わせに関するループとし、

前記最も外側のループの内側の２番目は前記縮約シェルＳに関するループ、前記２番目よりも内側の３番目は前記縮約シェルＵに関するループとするか、あるいは前記２番目は前記縮約シェルＵに関するループ、前記３番目は前記縮約シェルＳに関するループとし、

前記縮約シェルＳのとりうる値の範囲をＲ以下とし、

前記縮約シェルUのとりうる値の範囲をR以下とし、
前記３番目のループの内側で、所定の２電子積分の計算およびその結果を用いた所定のフォック行列要素の一部の計算を行うものであって、
前記２番目および３番目のループをひとまとまりとして１つのジョブ単位を形成し、前記ジョブ単位毎に前記プロセッサに処理を割り当てる並列処理方法において、
前記ジョブは、前記縮約シェルRの軌道量子数が高いジョブは、小さい番号を割り当て、前記縮約シェルRの軌道量子数が低いジョブには大きい番号を割り当てて形成することを特徴とする並列処理方法。

【請求項１３】請求項１２に記載の並列処理方法において、
請求項１、請求項２、請求項３、請求項５のいずれかに記載のジョブの割り当て方法を用いて、前記複数のプロセッサへの前記ジョブの割り当てを行うことを特徴とする並列処理方法。

【請求項１４】請求項１２に記載の並列処理方法において、
軌道量子数が高い縮約シェルには、小さい縮約シェル番号を割り当て、前記軌道量子数が低い縮約シェルには、大きい縮約シェル番号を割り当てて前記ジョブの番号を、R（R＋1）／２＋Tの算出式により定めて、N（Nは整数）からM（Nより大きい整数）まで連続する整数で番号付けを行い、
請求項４、請求項６、請求項７、請求項８、請求項９、請求項１０、請求項１１のいずれかに記載のジョブの割り当て方法を用いて、前記複数のプロセッサへの前記ジョブの割り当てを行うことを特徴とする並列処理方法。

【発明の詳細な説明】

【０００１】

【発明の属する技術分野】この発明は、ホスト計算機と、このホスト計算機と共通のバスを通じて接続される複数のプロセッサにより構成される計算機システムを用いて、複数個のジョブを、複数個のプロセッサにより並列に処理する並列処理方法におけるジョブの割り当て方法に関する。また、大規模で特定の対称性を有する行列要素計算、特に非経験的分子軌道法を用いた分子シミュレーションにおいてフォック行列要素計算を高速に処理するための並列処理方法に関する。

【０００２】

【従来の技術】化学の分野において、分子の状態や挙動を数値的に解析する手法として、分子軌道法、分子動力学法、モンテカルロ法などがある。その中でも、非経験的分子軌道計算法は、第一原理に基づいた量子力学的計算で、分子中の電子の挙動を記述することを目的としている。そのため、この手法は、分子シミュレーションの基盤として位置づけられ、物質構造や化学反応の詳細な解析に用いられている工業的に重要な手法である。

【０００３】非経験的分子軌道計算法では、分子を構成する原子の原子核と軌道電子との距離の２乗に経験的な定数を乗じたものを指数とする指数関数の逆数あるいはそれらの線形結合を基底関数とし、この基底関数を１つの原子に対して複数個用意する。これらの基底関数の線形結合で、分子内の軌道電子の波動関数、すなわち分子軌道を記述する。

【０００４】分子軌道における基底関数の線形結合係数を決めることが、非経験的分子軌道計算法での主要な処理であるが、その計算には、基底関数の数の４乗に比例した計算量と記憶容量を必要とする。そのため、非経験的分子軌道計算法は、現状では、１００原子程度の規模の分子系に適用されているに過ぎない。生命現象／化学現象の分子論的解明を、より現実的なものとするためには、数１０００原子規模の分子系への適用も視野に入れた、非経験的分子軌道計算専用の計算システムの開発が必須である。

【０００５】このように規模の大きい非経験的分子軌道計算専用の計算システムの例として、文献１（白川他“超高速分子軌道計算専用機ＭＯＥのアーキテクチャ”，電子情報通信学会技術報告，ｖｏｌ．ＣＰＳＹ９６－４６，ｎｏ．５，ｐｐ．４５－５０，１９９６）あるいは文献２（稲畑他，“ＰＰＲＡＭ－ＭＯＥ：分子軌道計算専用サーバＭＯＥのプロセッシング・ノードＬＳＩ”，電子情報通信学会技術報告，ｖｏｌ．ＣＰＳＹ９８－２１，ｎｏ．４，ｐｐ．７７－８４，１９９８）に記載のシステムがある。

【０００６】これらの文献に記載の計算システムは、ホスト計算機と、例えば１００個のプロセッサ（以下の説明においては、この並列処理のためのプロセッサを、プロセッサ・エレメントと呼ぶことにする。）とが、共通のバスにより接続された構成の並列処理システムで、非経験的分子軌道計算を高速に行うことを目的としている。

【０００７】これらのシステムは、個々の並列プロセッサ・エレメントが高い演算処理能力を有する上、システム全体を低価格で実現することが可能であるため、コストパフォーマンスに優れた計算システムを提供することができる。

【０００８】

【発明が解決しようとする課題】非経験的分子軌道計算のように、計算量が膨大なものである場合、並列処理用の複数個のプロセッサ・エレメントを効率的に運用して、より高速化することが重要である。特に、上記の文献に記載のシステムにおいては、各プロセッサ・エレメントは、共通のバスを通じてホスト計算機と通信を行って、必要なデータの授受を行いながら、計算処理を実行するので、できるだけ、各プロセッサ・エレメントにおける通信の待ち時間をなくして、効率的な処理を行えるようにすることが、高速処理のために重要である。

【０００９】しかしながら、単純に、非経験的分子軌道

計算について、ジョブを形成し、並列プロセッサ・エレメントに割り付けた場合には、ジョブ毎に、計算量（計算時間や通信量）が異なるため、通信のオーバーヘッド（待ち時間）が生じることにより、プロセッサ・エレメントの個数に比例した処理性能の向上が得られなくなる問題がある。

【００１０】並列処理を行うプロセッサ・エレメントの個数が増加すると、ホスト計算機とプロセッサ・エレメントとの間で送受信されるトータルのデータ量は変わらないが、プロセッサ・エレメントによる処理能力が個数に比例して増加する。その結果、一定時間内で行わなければならない通信の量（平均通信量）が増加し、プロセッサ・エレメント数が、ある個数を超えると平均通信量が通信網の処理能力（通信性能）を超えてしまう。このことが、通信のオーバーヘッドを発生させる一因となる。

【００１１】また、短い時間内に通信要求が集中的に発生し、合計の通信量が一時的に通信網の通信性能を超えてしまうような、時間的に不均一に通信負荷が発生するような場合にも、通信のオーバーヘッドが発生し、並列化効率を劣化させる原因となる。これは、平均通信量が通信網の通信性能を超えない場合であっても起こり得る。

【００１２】したがって、発生するトータルの通信量を少なくすることと、通信負荷の発生を時間的に均一にすること（通信負荷の時間的な分散）が、並列計算機で効率的な処理を行うために不可欠である。

【００１３】例えば、複数個のジョブのそれぞれが、計算量（計算時間；なお、この計算には、例えばＲＯＭを用いた変換処理なども含まれるものとする。この明細書では、計算と演算とは同義のものとして使用している）と通信量（通信時間）とが均等なジョブであれば、共通のバスを通じて行う通信タイミングを遅延させるように、各プロセッサ・エレメントにジョブを割り振ることにより、通信のオーバーヘッドを生じることなく、並列処理が行える。

【００１４】文献３（特開平５－３２４５８１号公報）には、多体問題の計算を並列処理する場合に計算負荷を均一化する負荷分散方法が開示されている。この文献３に記載の負荷分散方法では、各ジョブのサイズが全て等しいことを前提としており、プロセッサ・エレメントに割り当てるジョブの個数を均等にすることにより、計算負荷を均等にしている。

【００１５】しかし、非経験的分子軌道計算においては、各ジョブ毎に、計算ループの回数が異なり、また、ホスト計算機から取得するデータ量の違いによる通信量も異なる。このため、ジョブの個数を均等にするだけでは計算負荷を均等にすることができない問題がある。

【００１６】上記の文献の他にも、文献４（特開平６－３５８７０号公報）、文献５（特開平６－２４３１１２号公報）、文献６（特開平７－２９５９４２号公報）、文献８（特開平８－１８５３７７号公報）、文献９（特開平１０－２４０６９８号公報）などにも、複数のプロセッサ・エレメントにより並列処理を行う場合の負荷分散方法が開示されている。

【００１７】しかし、これら文献４、文献５、文献６、文献７、文献９に記載の負荷分散方法では、いずれもプロセッサ・エレメントの負荷の状態を測定あるいは予測した上でジョブのスケジューリングを行うものであるので、測定あるいは予測のための専用の機能を設けなければならない上に、測定あるいは予測のために付加的な処理が必要となるという問題があった。

【００１８】また、文献８に記載の分散計算システムでは、分散システムを均等な能力を有するグループに分け、グループ単位で負荷分散を行うようにすることが開示されているが、通信負荷の分散に関しては考慮されていないため、通信のオーバーヘッドが大きくなってしまうおそれがあるという問題がある。

【００１９】この発明は、上述の点にかんがみ、ジョブ毎に計算負荷や通信負荷の大きさが異なる場合においても、専用の機構を設けずに、簡単な方法で、負荷を均等化できるジョブの割り当て方法を提供して、効率的な並列処理を可能にし、システム全体の処理時間を短縮化できるようにすることを目的とする。

【００２０】

【課題を解決するための手段】上記課題を解決するために、請求項１の発明の並列処理方法におけるジョブの割り当て方法は、ホスト計算機と、このホスト計算機と共通のバスを通じて接続される複数のプロセッサにより構成される計算機システムを用いて、前記ホスト計算機と前記プロセッサとの間の通信処理と前記プロセッサにおける演算処理とからなる複数個のジョブを、前記複数個のプロセッサにより並列に処理する並列処理方法において、前記プロセッサのそれぞれには、前記ジョブの通信時間と演算時間との比の平均値が、所定の値になるように各ジョブを割り当てることを特徴とする。

【００２１】請求項２の発明は、請求項１に記載の並列処理方法におけるジョブの割り当て方法において、前記複数個のジョブの全体の通信時間と、前記複数個のジョブを前記複数個のプロセッサに均等に割り当てたとしたときの一つのジョブにおける演算時間の総計とが、ほぼ等しい時には、前記プロセッサのそれぞれには、前記ジョブの通信時間と演算時間との比の平均値が、前記プロセッサの数分の１になるように各ジョブを割り当てることを特徴とする。

【００２２】また、請求項３の発明は、請求項１または請求項２に記載の並列処理方法におけるジョブの割り当て方法において、各プロセッサには、通信時間と演算時間との比が、前記所定の値あるいは前記プロセッサの数分の１よりも大きいジョブと、小さいジョブとを交互に

割り当てることを特徴とする。

【００２３】また、請求項４の発明は、ホスト計算機と、このホスト計算機と共通のバスを通じて接続される複数のプロセッサにより構成される計算機システムを用いて、前記ホスト計算機と前記プロセッサとの間の通信処理と前記プロセッサにおける演算処理とからなるジョブの複数個を、前記複数個のプロセッサにより並列に処理する並列処理方法において、前記複数個のジョブには、Ｎ（Ｎは整数）からＭ（Ｎより大きい整数）まで連続する整数で番号付けを行い、直前に割り当ての済んだジョブの番号Ｊが（Ｎ＋Ｍ）／２以下であれば、番号がＮ＋（Ｍ－Ｊ－１）のジョブを次のジョブとして割り当て、直前に割り当ての済んだジョブの番号Ｊが（Ｎ＋Ｍ）／２より大きいならば、番号がＮ＋（Ｍ－Ｊ）のジョブを次のジョブとして割り当てるまたは、直前に割り当ての済んだジョブの番号Ｊが（Ｎ＋Ｍ）／２より小さいならば、番号がＮ＋（Ｍ－Ｊ）のジョブを次のジョブとして割り当て、直前に割り当ての済んだジョブの番号Ｊが（Ｎ＋Ｍ）／２以上であれば、番号がＮ＋（Ｍ－Ｊ－１）のジョブを次のジョブとして割り当てて、前記通信時間と前記演算時間との比の平均値が、所定の値になるように各ジョブを割り当てることを特徴とする。

【００２４】また、請求項５の発明は、ホスト計算機と、このホスト計算機と共通のバスを通じて接続される複数のプロセッサにより構成される計算機システムを用いて、前記ホスト計算機と前記プロセッサとの間の通信処理と前記プロセッサにおける演算処理とからなるジョブの複数個を、前記複数個のプロセッサにより並列に処理する並列処理方法において、前記複数個のプロセッサを、グループ内の前記プロセッサの数がほぼ等しいＧ個（Ｇは２以上の整数）のプロセッサ・グループに分割すると共に、前記複数個のジョブを、グループ内の前記ジョブの数がほぼ等しいＧ個のジョブ・グループに分割し、前記プロセッサ・グループと、前記ジョブ・グループとを１対１に対応させて、一つのジョブ・グループ内のジョブを、対応する一つのプロセッサ・グループ内のプロセッサに割り当てるものであって、前記通信時間と演算時間とからなるサイズと、前記通信時間と前記演算時間との比とが近似するジョブは、前記ジョブ・グループの異なるものにそれぞれ属するようにすると共に、前記通信時間と演算時間とからなるサイズと、前記通信時間と前記演算時間との比とが近似するジョブの、前記ジョブ・グループのそれぞれ内での割り当て順番を、前記複数のジョブ・グループにおいて互いに異なるように割り当てることを特徴とする。

【００２５】また、請求項１２の発明は、ｎ個（ｎは正の整数）の縮約シェルを用いて表現される分子のエネルギーを計算する分子軌道計算を、ホスト計算機と複数個のプロセッサとにより構成される計算機システムを用いて行う並列処理方法であって、縮約シェルＲ，Ｓ，Ｔ，Ｕのそれぞれに含まれる原始シェルｒ，ｓ，ｔ，ｕのそれぞれの成分である原始基底関数ｉ，ｊ，ｋ，ｌをインデックスとして用いて表わされる２電子積分関数ｇの関数値ｇ（ｉ，ｊ，ｋ，ｌ）と；前記原始基底関数ｋをひとつの構成要素とする縮約基底関数Ｋおよび前記原始基底関数ｌをひとつの構成要素とする縮約基底関数Ｌとをインデックスとして用いて表わされる密度行列Ｐの要素Ｐ（Ｋ，Ｌ）と；係数Ａ１との積Ａ１・Ｐ（Ｋ，Ｌ）・ｇ（ｉ，ｊ，ｋ，ｌ）の全ての縮約基底関数に関する総和ｆ１（Ｉ，Ｊ）と、前記２電子積分関数の関数値ｇ（ｉ，ｋ，ｊ，ｌ）と；前記密度行列Ｐの要素Ｐ（Ｋ，Ｌ）と；係数Ａ２との積Ａ２・Ｐ（Ｋ，Ｌ）・ｇ（ｉ，ｋ，ｊ，ｌ）の全ての縮約基底関数に関する総和ｆ２（Ｉ，Ｊ）との和ｆ（Ｉ，Ｊ）＝ｆ１（Ｉ，Ｊ）＋ｆ２（Ｉ，Ｊ）の、前記原始基底関数ｉ，ｊをひとつの構成要素とする前記縮約基底関数Ｉ，Ｊに含まれる全ての前記原始基底関数に関する和で表わされるフォック行列Ｆの全ての行列要素Ｆ（Ｉ，Ｊ）の計算を、最も外側のループは、Ｒ≦ｎmax（ｎmax はｎ個の縮約シェルに付与された番号の最大値）およびＴ≦Ｒなる関係を満たす前記縮約シェルＲとＴとの組み合わせに関するループとし、前記最も外側のループの内側の２番目は前記縮約シェルＳに関するループ、前記２番目よりも内側の３番目は前記縮約シェルＵに関するループとするか、あるいは前記２番目は前記縮約シェルＵに関するループ、前記３番目は前記縮約シェルＳに関するループとし、前記縮約シェルＳのとりうる値の範囲をＲ以下とし、前記縮約シェルＵのとりうる値の範囲をＲ以下とし、前記３番目のループの内側で、所定の２電子積分の計算およびその結果を用いた所定のフォック行列要素の一部の計算を行うものであって、前記２番目および３番目のループをひとまとまりとして１つのジョブ単位を形成し、前記ジョブ単位毎に前記プロセッサに処理を割り当てる並列処理方法において、前記ジョブは、前記縮約シェルＲの軌道量子数が高いジョブには小さい番号を割り当て、前記縮約シェルＲの軌道量子数の低いジョブには大きい番号を割り当てて形成することを特徴とする。

【００２６】また、請求項１３の発明は、請求項１２に記載の並列処理方法において、請求項１、請求項２、請求項３、請求項５のいずれかに記載のジョブの割り当て方法を用いて、前記複数のプロセッサへの前記ジョブの割り当てを行うことを特徴とする。

【００２７】また、請求項１４の発明は、請求項１２に記載の並列処理方法において、軌道量子数が高い縮約シェルには、小さい縮約シェル番号を割り当て、前記軌道量子数が低い縮約シェルには、大きい縮約シェル番号を割り当てて前記ジョブの番号を、Ｒ（Ｒ＋１）／２＋Ｔの算出式により定めて、Ｎ（Ｎは整数）からＭ（Ｎより大きい整数）まで連続する整数で番号付けを行い、請求項４、請求項６、請求項７、請求項８、請求項９、請求

項１０、請求項１１のいずれかに記載のジョブの割り当て方法を用いて、前記複数のプロセッサへの前記ジョブの割り当てを行うことを特徴とする。

【００２８】

【作用】上述の構成の請求項１の発明によれば、ジョブの通信時間と演算時間との比の平均値が、所定の値になるように、複数のプロセッサにジョブが割り当てられるので、通信処理も演算処理も効率的に行われ、処理終了までの時間を短縮化することができる。

【００２９】また、請求項２の発明によれば、複数個のジョブの全体の通信時間と、複数個のジョブを複数個のプロセッサに均等に割り当てたとしたときの一つのジョブにおける演算時間の総計とが、ほぼ等しい場合に、ジョブの通信時間と演算時間との比の平均値が、プロセッサの数分の１になるようにされるので、各プロセッサにおいて通信も演算も行っていない時間帯が生じることがなく、並列の複数個のプロセッサのそれぞれで前記演算時間が終了した時には、処理が終了となる。したがって、通信処理も演算処理も効率的に行われ、処理終了までの時間を短縮化することができる。

【００３０】請求項３の発明によれば、通信時間と演算時間との比が、所定の値あるいはプロセッサの数分の１よりも大きいものと、小さいものとを交互に割り当てるという簡単な方法により、通信処理と演算処理とを効率的に行うことができるようになる。

【００３１】請求項４の発明によれば、ジョブには、通信時間と演算時間との比の小さいものから順に、あるいは大きいものから順に、Ｎ（Ｎは整数）からＭ（Ｎより大きい整数）まで連続する整数で番号付けが行われている。そして、複数個のジョブのサイズは、ばらつきが小さいものとされている。

【００３２】そして、プロセッサにおける前処理のジョブ番号が、全体のジョブ番号のなかの中間値よりも大きいときには、次のジョブとしては、ジョブ番号が小さいものが割り当てられ、前処理のジョブ番号が、全体のジョブ番号のなかの中間値よりも小さいときには、次のジョブとしては、ジョブ番号が大きいものが割り当てられる。

【００３３】したがって、ジョブの通信時間と演算時間との比の平均値が、ほぼプロセッサの数分の１になるようにされ、通信処理も演算処理も効率的に行われ、処理終了までの時間を短縮化することができる。

【００３４】また、請求項５の発明においては、複数個のプロセッサは、グループ内のプロセッサの数がほぼ等しいＧ個（Ｇは２以上の整数）のプロセッサ・グループに分割される。また、複数個のジョブは、グループ内のジョブの数が、プロセッサ・グループの数に等しいＧ個のジョブ・グループに分割される。そして、プロセッサ・グループと、ジョブ・グループとが１対１に対応させられて、一つのジョブ・グループ内のジョブが、対応する一つのプロセッサ・グループ内のプロセッサに割り当てられる。

【００３５】この場合に、通信時間と演算時間とからなるサイズと、通信時間および演算時間が近似するジョブは、ジョブ・グループの異なるものにそれぞれ属するように分散され、しかも、通信時間と演算時間とからなるサイズと、通信時間および演算時間が近似するジョブの、ジョブ・グループのそれぞれ内での割り当て順番が、複数のジョブ・グループにおいて互いに異なるように割り当てられて分散され、複数個のジョブによる通信処理が共通バス上で輻輳状態にならないようにされる。

【００３６】したがって、通信処理も演算処理も効率的に行われ、処理終了までの時間を短縮化することができる。

【００３７】また、請求項１２の発明は、非経験的分子軌道計算におけるフォック行列計算アルゴリズムによるフォック行列の計算を、ホスト計算機とプロセッサ・エレメントとの間の通信性能およびホスト計算機の処理性能に律速されることなく、高速に行うことができる並列処理方法である。

【００３８】この場合、縮約シェルＲおよびＴに関するループはホスト計算機で制御され、ＲとＴの組み合わせの固定されたＳおよびＵに関するループの部分がプロセッサ・エレメントに割り当てられる。

【００３９】Ｒ≧Ｔ，Ｓ，Ｕの関係があるので、Ｒの番号の大きいものは、ループ回数も大きい。一方、Ｒの番号が小さいものは、ループ回数が少ない。また、軌道量子数が高いシェルは、計算に必要な密度行列が多数必要になり、その分、通信量が大きくなり、軌道量子数が低いシェルは、密度行例が少なく、その分、通信量が少なくなる。

【００４０】このため、この請求項１２のように、縮約シェルＲの軌道量子数が高いジョブには小さいジョブ番号が割り当てられ、縮約シェルＲの軌道量子数が低いジョブには大きいジョブ番号が割り当てられると、演算処理と通信処理との和からなるジョブのサイズのばらつきが小さくなる。

【００４１】したがって、並列処理の複数のプロセッサ・エレメントへのジョブの、効率的な割り当てを容易に行えるようになる。

【００４２】そこで、請求項１３の発明のように、請求項１２の分子軌道計算の並列処理方法において、請求項１、請求項２、請求項３、請求項５のいずれかに記載のジョブの割り当て方法を用いて、前記複数のプロセッサへの前記ジョブの割り当てを行うことにより、通信処理も演算処理も効率的に行われ、処理終了までの時間を短縮化することができる。

【００４３】また、請求項１４の発明の場合には、軌道量子数の高い縮約シェルには、小さい縮約シェルの番号を割り当て、軌道量子数の低い縮約シェルには、大きい

縮約シェルの番号を割り当て、R（R＋1）／2＋Tの算出式により定めて、N（Nは整数）からM（Nより大きい整数）まで連続する整数で番号付けを行うので、結果として得られるジョブ番号は、通信時間と演算時間の比の大きさに概略したがった順番となる。

【００４４】このため、請求項１２の分子軌道計算の並列処理方法において、請求項４、請求項６、請求項７、請求項８、請求項９、請求項１０、請求項１１のいずれかに記載のジョブの割り当て方法を用いて、前記複数のプロセッサへの前記ジョブの割り当てを行うことにより、通信処理も演算処理も効率的に行われ、処理終了までの時間を短縮化することができる。

【００４５】

【発明の実施の形態】以下、この発明による並列処理方法およびそのジョブの割り当て方法を、非経験的分子軌道計算法におけるフォック行列の計算アルゴリズムを行う場合に適用した実施の形態について、図を参照しながら説明する。

【００４６】この発明による並列処理方法およびそのジョブの割り当て方法の実施の形態を説明する前に、非経験的分子軌道法、それに用いる計算機システムの構成例および、その非経験的分子軌道法におけるフォック行列の新規な計算アルゴリズムについて説明する。

【００４７】［非経験的分子軌道法の概要］非経験的分子軌道計算法では、分子の状態を表わす波動関数Ψを、分子中の電子の空間的な軌道に相当する電子軌道関数ϕ_μを用いて記述する。ここでμは複数ある分子軌道のμ番目という意味の添え字である。分子軌道ϕ_μは、原子軌道χ_Iの線形結合で、図３３の（数式１）のように近似的に表わされる。

【００４８】ここで、（数式１）において、Ｉは複数ある原子軌道のＩ番目という意味の添え字である。なお、原子軌道は基底関数とも呼ばれることがある。この明細書中では、以降、原子軌道のことを基底関数と呼ぶ。また、（数式１）に現れる$C_{I\mu}$は、線形結合係数である。（数式１）におけるＩに関する総和は、計算の対象とする分子を構成する全ての基底関数に関するものである。

【００４９】さて、量子力学的に分子軌道を記述するためには、良く知られるパウリの排他律を、分子内の電子の状態が満たさなければならない。電子のスピンを考慮に入れて、パウリの排他律を満たすように、２ｎ電子系の分子の状態Ψを記述する表式として、図３３の（数式２）のようなスレーター行列式が用いられる。ここで、（数式２）において、α（ｘ）およびはβ（ｘ）、ｘ番目の電子のスピンが、それぞれ、上向きおよび下向きの状態を表わしている。

【００５０】２ｎ電子系に対するハミルトニアンＨは、１電子部分H_1と２電子部分H_2との和という形式で、図３３の（数式３）から（数式５）のように書き表される。

【００５１】図２３の（数式４）において、右辺の（・・・）内の第１項は、電子ｐの運動エネルギー、第２項はｐ番目の電子とＡ番目の原子核との相互作用である。（数式４）において、Σ_p（この明細書でΣ_iは、ｉについての総和を取ることを表すものとする。以下、同じ）は全電子に関する総和、Σ_Aは全原子核に関する総和、Z_Aは原子核Ａの電荷、r_{pA}は電子ｐと原子核Ａとの距離である。

【００５２】また、（数式５）は、電子ｐと電子ｑとの間の相互作用を表わしており、$\Sigma_p\Sigma_{q(>p)}$は２個の電子の組み合わせに関する総和、r_{pq}は電子ｐ，ｑ間の距離である。

【００５３】上記のハミルトニアンＨと、（数式２）のスレーター行列式とを用いると、分子エネルギーの期待値εが、図３４の（数式６）～（数式９）のように表わされる。

【００５４】（数式６）において、Σ_μおよびΣ_νは、ｎ個（ｎは正の整数）ある分子軌道に関する総和である。（数式７）は「コア積分」と呼ばれ、代表として番号１の電子について書かれている。また、（数式８）および（数式９）は、それぞれ「クーロン積分」および「交換積分」と呼ばれ、代表として電子１および電子２について書かれている。なお、（数式６）から（数式９）において、$d\tau$，$d\tau_1$，$d\tau_2$は、位置座標空間での体積積分を表わす。

【００５５】（数式６）を基底関数を用いて書き直すと、図３５に示す（数式１０）～（数式１３）に示すようなものになる。（数式１３）で表わされる積分を、「２電子間反発積分」あるいは省略して「２電子積分」と呼ぶ。

【００５６】（数式１０）で表わされる分子エネルギーの期待値εは、$C_{I\mu}$という未知数を含んでおり、このままでは数値が得られない。$C_{I\mu}$は、（数式１）における線形結合定数であり、μは１からｎ（分子軌道の数）の整数、Ｉは１からＮ（Ｎが基底関数の数であり、正の整数）の整数である。以下では、$C_{I\mu}$を要素とするＮ×ｎ行列Ｃを係数行列と呼ぶ。

【００５７】期待値εが最小となるように係数行列を決定し、基底状態の波動関数Ψを求める手法の１つとして、ハートリー・フォック・ローサーンの変分法（以下、ＨＦＲ法と略称する）が用いられる。導出過程は省略し、ＨＦＲ法の結果として得られる式を、図３６の（数式１４）～（数式１８）に示す。

【００５８】F_{IJ}はフォック行列要素、P_{KL}は密度行列要素と、それぞれ呼ばれる。以下の説明では、これらをＦ（Ｉ，Ｊ）、Ｐ（Ｋ，Ｌ）のように表記する場合がある。これらは、１からＮの値をとる各Ｉ，Ｊ，Ｋ，Ｌに対して数値を持っており、それぞれＮ×Ｎ行列の形で表わされる。

【００５９】（数式１４）を解くことにより、係数行列が求まる。（数式１４）は、１からｎの間の全てのμ、および１からＮの間の全てのＩに対して存在するので、ｎ×Ｎ本の連立方程式になっている。

【００６０】（数式１４）を解いて得られた係数行列Ｃの計算には、密度行列Ｐが用いられている。密度行列Ｐは、（数式１８）に示すように係数行列Ｃから計算される。そのため、具体的な計算手順としては、まず、適当に係数行列Ｃを与えておき、それを用いて計算した密度行列Ｐを使って、（数式１５）でフォック行列Ｆを計算し、（数式１４）の連立方程式を解いて新たな係数行列Ｃを得る。密度行列Ｐの元となるＣと、結果として得られるＣとの間の差が十分小さく、すなわち自己無撞着になるまで、上記の計算を繰り返し行う。この反復計算を自己無撞着計算（以下、ＳＣＦ計算と称する）と呼ぶ。

【００６１】実際の計算で最も時間を要するのは、（数式１５）のフォック行列要素F_{IJ}の計算である。これは、全てのＩ，Ｊに対して、この（数式１５）を計算しなければならないこと、および各Ｉ，Ｊの組み合わせに対して、密度行列要素P_{KL}のＫ，Ｌに関する和を計算しなければならないことに起因する。

【００６２】ＳＣＦ計算の手法には２通りある。１つはディスクストレージＳＣＦ法と呼ばれる手法で、１回目のＳＣＦ計算の際に得た２電子積分の値を全てディスクに保存しておき、２回目以降は必要な２電子積分をディスクから取り出して用いる手法である。もう１つはダイレクトＳＣＦ法と呼ばれる手法で、ＳＣＦ計算の度に２電子積分の計算をやり直す手法である。

【００６３】現在では、ディスク容量の制限やアクセス時間の大きさなどから、後者のダイレクトＳＣＦ法を用いるのが主流である。このダイレクトＳＣＦ法による分子軌道計算においては、ＳＣＦ計算の１回あたりに、N^4にほぼ比例する個数の２電子積分の計算を行わなければならないため、２電子積分計算を高速に行うことが分子軌道計算を高速化することに直結する。

【００６４】２電子積分Ｇ（Ｉ，Ｊ，Ｋ，Ｌ）、密度行列Ｐ（Ｋ，Ｌ）、およびフォック行列Ｆ（Ｉ，Ｊ）の対称性に関して、ここで言及しておく。

【００６５】２電子積分は、（数式１３）から明らかなように、図２６の（数式１９）に示すような対称性を有している。したがって、（数式１９）の内の１つに関して数値を得ることができれば、他の７つについても数値が得られたことになる。

【００６６】また、図３６の（数式１８）から、

Ｐ（Ｋ，Ｌ）＝Ｐ（Ｌ，Ｋ）

であることがわかり、図２６の（数式１５）および図２５の（数式１１）から、

Ｆ（Ｉ，Ｊ）＝Ｆ（Ｊ，Ｉ）

であることがわかる。

【００６７】［縮約基底関数と原始基底関数］非経験的分子軌道法では、図３７の（数式２０）に示すような基底関数が一般的に用いられる。この（数式２０）において、ｒ，ｎ，Ｒはベクトルであり、添え字ｘ，ｙ，ｚの付いたものがその成分である。ｒは電子の座標、ｎは電子の角運動量、Ｒは原子核の座標である。

【００６８】$n_x+n_y+n_z=\lambda$は、角運動量の大きさであり、軌道量子数とも呼ばれる。この軌道量子数λが、０の場合にその軌道をｓ軌道、１の場合にその軌道をｐ軌道、２の場合にその軌道をｄ軌道などと呼ぶ。

【００６９】ζ_mは軌道指数であり、軌道の空間的な広がり具合を示す。軌道指数の異なる複数の軌道の線形結合で１つの基底関数を表わす場合があり、そのようにして表わした基底関数を縮約基底関数と呼び、線形結合係数d_mを縮約係数と呼ぶ。これに対して、線形結合される前の、図２７の（数式２１）の形の関数ψを原始基底関数と呼ぶ。

【００７０】縮約基底関数χは、Ｉ，Ｊ，Ｋ，Ｌのように大文字で番号付けをし、また、原始基底関数ψは、ｉ，ｊ，ｋ，ｌのように小文字で番号付けするのが慣例であり、本明細書中でもこれに従う。

【００７１】［縮約シェルと原始シェル］軌道量子数が１の場合の縮約基底関数には、ｎ＝（１，０，０）の場合、ｎ＝（０，１，０）の場合、ｎ＝（０，０，１）の場合の３通りが存在する。同様に、軌道量子数が２の場合には６通り（あるいは、基底関数の構成の仕方によっては５通り）の縮約基底関数が存在する。

【００７２】（数式２０）のうちの図３７の（数式２２）で示す部分が共通な、これら複数の縮約基底関数の集合を、縮約シェルと呼ぶ。ｐ軌道の縮約シェルは３つの縮約基底関数で構成され、また、ｄ軌道の縮約シェルは６つ（または５つ）の縮約基底関数で構成される。ｓ軌道の場合にも、便宜上１つの縮約基底関数の集合を縮約シェルと呼ぶ。

【００７３】（数式２１）のうちのｅｘｐ［－ζ（ｒ－Ｒ）^{2}］の部分が共通な、原始基底関数の集合を、同様に原始シェルと呼ぶ。縮約シェルは、Ｒ，Ｓ，Ｔ，Ｕのように大文字で番号付けをし、原始シェルは、ｒ，ｓ，ｔ，ｕのように小文字で番号付けするのが慣例であり、本明細書中でもこれに従う。

【００７４】分子軌道計算の実施に際しては、計算の対象とする分子を構成する原子毎に軌道量子数の異なる複数の縮約シェルを用意し、それら全ての集合を基底関数のセットとして用いる。原子核座標Ｒと軌道量子数λとの組み合わせ（Ｒ，λ）で、１つの縮約シェルを表わすことができる。

【００７５】［２電子積分の表式］縮約基底関数で表わされる２電子積分Ｇ（Ｉ，Ｊ，Ｋ，Ｌ）は、原始基底関数を用いると、図３７の（数式２３）のように表わされる。ここで、ｇ（ｉ，ｊ，ｋ，ｌ）は、図３７の（数式２４）のように表すことができる。

【００７６】Ｇ（Ｉ，Ｊ，Ｋ，Ｌ）を、縮約基底関数で表現した２電子積分と呼び、ｇ（ｉ，ｊ，ｋ，ｌ）を、原始基底関数で表現した２電子積分と呼ぶが、以降の説明では、どちらも単に２電子積分と呼ぶ場合がある。ｇ（ｉ，ｊ，ｋ，ｌ）も、図３７の（数式２５）で示すような対称性を有している。

【００７７】さて、原始基底関数ψは、その角運動量ｎ、軌道指数ζ、原子核座標Ｒの組み合わせで、一意的に示すことができる。ｉ，ｊ，ｋ，ｌ番目の原始基底関数が、図３２に示す表１のような角運動量、軌道指数、原子核座標を有するものと仮定する。

【００７８】説明の煩雑さを避けるために、以下の説明では、原始基底関数の番号ｉ，ｊ，ｋ，ｌの代わりに、それぞれの角運動量ａ，ｂ，ｃ，ｄを用いて２電子積分を［ａｂ，ｃｄ］のように表わすことにする。

【００７９】上記のように用意された基底関数セットを用いて２電子積分を計算する効率的な手法を、文献１（Ｓ．Ｏｂａｒａ　ａｎｄ　Ａ．Ｓａｉｋａ，ＪＣＰ，ｖｏｌ．８４，ｎｏ．７，ｐ．３９６４，１９８６）に従って説明する。

【００８０】まず、ａ，ｂ，ｃ，ｄが全てｓ軌道、すなわち$a=0_a=(0,0,0)$，$b=0_b=(0,0,0)$，$c=0_c=(0,0,0)$，$d=0_d=(0,0,0)$である場合には、（数式２４）の２電子積分は、図３８の（数式２６）～（数式３４）に示すように求まる。

【００８１】ここで、（数式２６）に現れる$[\cdot\cdot,\cdot\cdot]^{(m)}$は補助積分、ｍは補助インデックスであるが、これらについては後で述べる。（数式２７）の積分範囲は、０から１である。

【００８２】また、ａ，ｂ，ｃ，ｄのうち１つでもｓ軌道以外のものがある場合には、図３９の（数式３５）および（数式３６）に示す漸化式を用いて計算する。

【００８３】（数式３５）で、添え字のｉは、ｘまたはｙまたはｚ成分であることを示す。また、1_iは、ｉ成分のみ１で、他は０であるようなベクトルである。さらに、$N_i(n)$は、角運動量ｎのｉ成分の値を示すものである。（数式３５）は、左辺の補助積分に現れる角運動量の１つは右辺では確実に１以上減少し、また、左辺の補助積分の補助インデックスは右辺では同じあるいは１だけ増加する、という性質を有している。

【００８４】補助積分$[\cdot\cdot,\cdot\cdot]^{(m)}$は、補助インデックスｍが０であるときに、２電子積分［・・，・・］に正確に一致するものであり、２電子積分の計算を補助するものである。どんなに角運動量の大きな基底関数を含んだ２電子積分であっても、（数式３５）を繰り返し用いて角運動量を減少させ、最後には、全て角運動量が（０，０，０）であるような補助積分に行き着くことができる。角運動量が（０，０，０）であるような補助積分は、（数式２６）を用いて計算できるので、その数値と適当な係数を乗じ、加算すれば２電子積分の数値が得られる。

【００８５】実際の計算は、以下のように行う。まず、（数式３５）に従って、２電子積分を８つ以下の補助積分を用いた形式に表わす。ここで現れた補助積分に対して、さらに、（数式３５）を適用する。このような手続きを繰り返して、全て角運動量が（０，０，０）であるような補助積分に行き着くまでの道筋を、計算手順として記録しておく。

【００８６】次に、（数式２６）を用いて角運動量が（０，０，０）であるような補助積分を計算し、そこから出発して、先ほどの計算手順をたどりながら補助積分の数値を計算していき、最後に目的とする２電子積分の数値を得る。

【００８７】（数式３５）が有するもう一つの重要な性質は、２電子積分に現れる４つの角運動量の組み合わせが同じであれば、軌道指数や原子核座標の組み合わせが異なっていても、上記の計算手順としては全く同じものを用いることができることである。計算の実行に際しては、軌道指数や原子核座標に応じて補助積分に乗じる係数を変えてやるだけで良い。

【００８８】［カットオフ］上述したように、計算しなければならない縮約基底関数で表わした２電子積分の数は、縮約基底関数の数Ｎに対してN^4となる。実際に数値を得なければならないのは、原始基底関数で表わした２電子積分の方であるが、その総数は、縮約基底関数で表わした２電子積分の数の数倍から数１０倍（縮約基底関数を構成する原始基底関数の数、すなわち縮約数に依存する）に及ぶ。

【００８９】この個数を減らす手法として、第１に考えられるのは、（数式１９）あるいは（数式２５）に記した対称性を利用することである。しかしながら、この方法では最も効率化を行っても２電子積分の計算量は１／８にしかならない。

【００９０】もう１つの手法は、計算精度の観点から、不必要と判断できる２電子積分の計算を、積極的に排除する方法である。不必要な２電子積分の判断は、以下のように行うことができる。

【００９１】上述したように、全ての２電子積分の数値は、（数式２６）に示した全て角運動量が（０，０，０）であるような補助積分$[00,00]^{(m)}$の数値に基づいて計算される。したがって、２電子積分の数値の、フォック行列要素の数値への寄与が計算誤差の程度であるかどうかを、$[00,00]^{(m)}$の数値で判断することが可能である。さらに、$[00,00]^{(m)}$の数値の大きさは、（数式２９）に示した関数Ｋ（ζ，ζ'，Ｒ，Ｒ’）の値から、さらに、それは、図３９の（数式３７）の大きさから判断することができる。

【００９２】したがって、（ζ_a，Ａ，ζ_b，Ｂ）の数値の組み合わせで、（数式２６）の１つ目の関数Ｋの大

きさを見積ることで、２電子積分［ａｂ，＊＊］を計算する必要があるかどうかを判断し、また、（ζｃ，Ｃ，ζｄ，Ｄ）の数値の組み合わせで、（数式２６）の２つ目の関数Ｋの大きさを見積ることで、２電子積分［＊＊，ｃｄ］を計算する必要があるかどうかを判断することができる。

【００９３】このようにして、不必要な２電子積分の計算を排除することを、「カットオフする」と呼ぶことにする。上記の例で、ａおよびｂの情報だけから判断してカットオフする場合には、ａｂでのカットオフ、ｃおよびｄの情報だけから判断してカットオフする場合には、ｃｄでのカットオフ、と呼ぶ場合がある。このように、ａｂだけで、あるいはｃｄだけでカットオフするかどうかの判断ができるのは、図２９の（数式３７）の最大値が１で下限値が０だからである。このようにカットオフを行うことにより、計算しなければならない２電子積分は、概略でN^2に比例する個数となり、計算量を大幅に低減できる。

【００９４】上述のことから、Ｎが大きい場合には、２電子積分の対称性を利用することによる効果よりも、カットオフによる計算量低減の効果の方が桁違いに大きく、これを取り入れることによって、非経験的分子軌道計算におけるフォック行列の計算に要する処理時間が大きく短縮できることがわかる。

【００９５】［分子軌道計算機システムの例］図１は、この実施の形態の方法が適用される並列計算システムの一例の全体の構成を示すもので、これは、先に説明した文献１に示されているものである。すなわち、このシステムは、ホスト計算機１に対して、共通のバス３を通じて、複数個のプロセッサ・エレメント２が接続されている。

【００９６】また、図２は、この実施の形態の方法が適用される並列計算システムの一例の全体の構成を示すもので、これは、先に説明した文献２に示されているものである。この図２の例のシステムでは、ホスト計算機１１に対して、シリアルバス１３により、ディジーチェーン型に複数個のボード１２が接続されている。バス１３としては、例えばＩＥＥＥ１３９４シリアルバスが用いられる。

【００９７】各ボード１２は、互いに数珠つなぎとされた複数個のプロセッサ・エレメント１４を備えると共に、バス１３とこれらの複数個のプロセッサ・エレメント１２との間の橋渡しを行うブリッジ１５を備えている。

【００９８】ところで、図１あるいは図２に示した、文献１あるいは文献２のシステムでは、個々の並列プロセッサ・エレメント２あるいは１４に接続できるメモリ容量に制限がある。現状では、各プロセッサ・エレメント２あるいは１４に、せいぜい数１０メガ・ビット（以下、Ｍｂと記す）程度の容量のメモリを接続するのが精一杯である。

【００９９】また、ホスト計算機１あるいは１１と、各プロセッサ・エレメント２あるいは１４とを接続する通信網として、上述のように、比較的汎用性が高く安価なＩＥＥＥ１３９４シリアルバスなどを用いた場合、その通信性能は、現状では、ピーク性能で、高々、４００メガ・ビット／秒（以下、Ｍｂｐｓと記す）程度である。

【０１００】この発明の実施の形態では、このような構成の計算機システムで、高い通信性能を必要とせず、なおかつ各プロセッシング・ノードすなわちプロセッサ・エレメントに接続されているメモリが、上記のような現実的な容量であっても、効率的に計算を行なえるような新規なフォック行列生成アルゴリズムを用いる。

【０１０１】［新規なフォック行列生成アルゴリズムの説明］効率的な並列計算を可能とするフォック行列生成アルゴリズムを、プログラム・コードの形式で示した図３を用いて説明する。

【０１０２】説明の便宜上、図３は、縮約シェルＲ，Ｓ，Ｔ，Ｕに関する４重ループと、その内部の縮約基底Ｉ，Ｊ，Ｋ，Ｌに関する４重ループにより構成されるように記しているが、実際のアルゴリズムでは、最も外側のループはＲ≦Ｎｍａｘ（ＮｍａｘはＮ個の縮約シェルに付与された番号の最大値である。０からＮまでの番号を付与した場合には、Ｎｍａｘ＝Ｎである。）およびＲ≧Ｔなる関係を満たす縮約シェルＲとＴとの組み合わせ（ＲＴ）に関するループとする。そして、２番目は縮約シェルＳに関するループとし、３番目は縮約シェルＵに関するループとする。３番目のループの内側で、関数値Ｇ（Ｒ，Ｓ，Ｔ，Ｕ）の計算およびその結果を用いた所定のフォック行列要素Ｆの一部の計算を行うようにする。

【０１０３】なお、２番目はＵに関するループ、３番目はＳに関するループとするようにしてもよい。すなわち、図３では、Ｕに関するループが、Ｓに関するループの内側に形成されているが、このループの順序は逆でも良い。

【０１０４】縮約シェルＲと縮約シェルＴとの組み合わせに関するループは、ホスト計算機上で制御され、縮約シェルＲと縮約シェルＴの組み合わせの固定された縮約シェルＳおよび縮約シェルＵに関するループの部分の処理がプロセッサ・エレメントに割り当てられる。ＲとＴとが固定された２番目および３番目の２つのループをひとまとまりとして一つのジョブ単位を形成し、このジョブ単位毎に複数のプロセッサ・エレメント２あるいは１４に処理させるようにする。

【０１０５】このように、複数のプロセッサ・エレメントへのジョブ割り当てが、ＲＴペア単位で行われることから、以下の説明においては、このアルゴリズムを、ＲＴ並列アルゴリズムと呼ぶことにする。

【０１０６】このとき、ホスト計算機１あるいは１１

は、全ての行列要素Ｐの初期値の計算と、プロセッサ・エレメントと共に行うＳＣＦ計算と、このＳＣＦ計算を継続するかどうかの決定などを行う。ＳＣＦ計算に当たって、ホスト計算機１あるいは１１は、複数個のプロセッサ・エレメント２あるいは１４に対するＲとＴとが固定されたジョブ単位の割り当て決定と、プロセッサ・エレメントに対して送信すべき行列Ｐの一部の行列要素の選択と、選択された行列要素のプロセッサ・エレメントに対する送信と、プロセッサ・エレメントから送信されてきた行列Ｆの一部の行列要素の受信と、行列Ｆを用いた行列Ｐの更新とを行う。

【０１０７】一方、プロセッサ・エレメント２あるいは１４は、ホスト計算機１あるいは１１から送信された行列Ｐの一部の行列要素の受信と、縮約シェルＳに関するループの制御および縮約シェルＵに関するループの制御と、関数Ｇ（Ｒ，Ｓ，Ｔ，Ｕ）または関数ｇ（ｉ，ｊ，ｋ，ｌ）の計算と、行列Ｆの一部の行列要素の計算と、行列Ｆの一部の行列要素のホスト計算機１あるいは１１に対する送信とを行う。

【０１０８】縮約シェルＲは０からＮｓｈｅｌｌ－１まで、縮約シェルＳ，Ｔ，Ｕは０からＲまでの値をとる。そして、ＩはＲに属する全ての縮約基底番号を、ＪはＳに属する全ての縮約基底番号を、ＫはＴに属する全ての縮約基底番号を、ＬはＵに属する全ての縮約基底番号を、それぞれとる。

【０１０９】なお、図３では、同じ縮約シェルに属するものが連続した番号となるように縮約基底が番号付けされていることを仮定して、Ｉ，Ｊ，Ｋ，Ｌの範囲を、それぞれｂ＿ｂａｓｉｓ（Ｒ）からｅ＿ｂａｓｉｓ（Ｒ）、ｂ＿ｂａｓｉｓ（Ｓ）からｅ＿ｂａｓｉｓ（Ｓ）、ｂ＿ｂａｓｉｓ（Ｔ）からｅ＿ｂａｓｉｓ（Ｔ）、ｂ＿ｂａｓｉｓ（Ｕ）からｅ＿ｂａｓｉｓ（Ｕ）としている。

【０１１０】ここで、ｂ＿ｂａｓｉｓ（Ｘ），ｅ＿ｂａｓｉｓ（Ｘ）は、縮約シェルＸに属する縮約基底の番号の最小値および最大値である。縮約基底に関する４重ループの内部では、２電子積分Ｇ（Ｉ，Ｊ，Ｋ，Ｌ）の計算、およびフォック行列要素の計算を行なう。ここで用いられる密度行列要素は、Ｐ（Ｋ，Ｌ），Ｐ（Ｉ，Ｊ），Ｐ（Ｋ，Ｊ），Ｐ（Ｉ，Ｌ）の４つであり、また、計算されるフォック行列要素は、Ｆ（Ｉ，Ｊ），Ｆ（Ｋ，Ｌ），Ｆ（Ｉ，Ｌ），Ｆ（Ｋ，Ｊ）の４つである。

【０１１１】ここで、ＩおよびＫは、それぞれ縮約シェルＲおよびＴを形成する縮約基底であるので、ジョブが割り当てられた時点で、これらは数個（縮約シェルがｓ軌道の場合には１個、ｐ軌道の場合には３個、ｄ軌道の場合には６個）に固定される。ＪおよびＬは任意である。

【０１１２】図３では、コードを簡潔に記述するために、ＳおよびＵが、０からＲまでのＲ＋１個の全ての値をとるような書き方をしているが、実際にＳおよびＵがその範囲の全ての値をとるわけではない。

【０１１３】Ｒに属する全ての縮約基底と、Ｓに属する全ての縮約基底との間で、abでのカットオフが成り立つ場合には、全てのＧ（Ｉ，Ｊ，＊，＊）の計算を省略できるので、Ｓはそのような値をとる必要がない。また、それに伴い、Ｐ（Ｉ，Ｊ），Ｐ（Ｋ，Ｊ）は用いられることがなく、Ｆ（Ｉ，Ｊ），Ｆ（Ｋ，Ｊ）は計算されることがないので、これらをホスト計算機とプロセッサ・エレメントとの間で通信する必要がない。

【０１１４】同様に、Ｔに属する全ての縮約基底と、Ｕに属する全ての縮約基底との間で、ｃｄでのカットオフが成り立つ場合には、全てのＧ（＊，＊，Ｋ，Ｌ）の計算を省略できるので、Ｕはそのような値をとる必要がない。また、それに伴い、Ｐ（Ｉ，Ｌ），Ｐ（Ｋ，Ｌ）は用いられることがなく、Ｆ（Ｉ，Ｌ），Ｆ（Ｋ，Ｌ）は計算されることがないので、これらをホスト計算機とプロセッサ・エレメントとの間で通信する必要がない。

【０１１５】したがって、ＳおよびＵに関するループを回る回数を削減して計算量を削減できるだけでなく、不必要な行列要素の通信をなくして通信量を削減することができる。

【０１１６】ＲＴ並列アルゴリズムを用いた処理の流れを、図４に示すフローチャートを用いて説明する。以下の説明では、図２のシステムを用いた場合とする。また、図４においては、ホスト計算機１１とプロセッサ・エレメント１４とでは異なる処理を行うため、それぞれについてのフローチャートを併記した。

【０１１７】また、プロセッサエレメント１４は、並列に複数（典型的には１００個）がホスト計算機１１に接続されていることを前提としているが、それらは同様のフローチャートに従って動作し、また、それらの間のデータの相関はないので、代表として１つのプロセッサエレメント１４における処理フローを表記した。なお、図４において、破線で示した矢印の部分は、その後の処理が、その前の処理で律速されるのではなく、他の処理系からの情報の入力待ちとなることを示している。

【０１１８】まず、ホスト計算機１１の処理手順を説明する。ホスト計算機１１は、係数行列の初期設定（ステップＳ１０１）を行った後、ＳＣＦループに入る。ＳＣＦループは、新たに計算された係数行列が自己無撞着となるまで継続する。

【０１１９】すなわち、ＳＣＦループ内では、まず、（数式１８）に従って係数行列から密度行列を計算する（ステップＳ１０２）。そして、ＲＴループに入り、まず、プロセッサ・エレメントに割り当てるＲＴペア番号、すなわち２つの縮約シェルＲとＴの組み合わせを一意的に表わす番号を決める（ステップＳ１０３）。この例の場合には、ＲＴペア番号は、図３９の（数式３８）

に示すような算出式により、定められる。

【０１２０】図３の記述に従うと、ＲＴペア番号は、小さい番号から順に割り当てられることになるが、必ずしも、上述のようにする必要はなく、全てのＲ、Ｔの組み合わせに対して処理が行なわれるようにすれば、どのような順序で処理を行なうように番号付けあるいは処理の順序付けをしても良い。

【０１２１】次に、上記のＲＴペア番号に対応したジョブで必要となる密度行列情報を、そのＲＴペア番号が割り当てられたプロセッサ・エレメントへ送信する（ステップ１０４）。このとき、ａｂおよびｃｄでのカットオフを考慮して、プロセッサ・エレメントにおける計算で用いられることのない密度行列データは一切送信しない。

【０１２２】プロセッサエレメント１４は、このホスト計算機１１からの密度行列情報を受信し、受信バッファメモリに格納する（ステップＳ２０１）。そして、割り当てられた（ＲＴペア）のジョブについて、縮約シェルＳ，Ｕに関するループを制御して、フォック行列要素の計算を行う（ステップＳ２０２）。そして、計算が終了すると、求めたフォック行列情報を送信バッファメモリに格納し、その送信バッファメモリからホスト計算機１１に、その求めたフォック行列情報を送信する（ステップＳ２０３）。

【０１２３】ホスト計算機１１は、以上のようにして、あるプロセッサ・エレメント１４での処理が終了して、そのプロセッサ・エレメント１４からフォック行列情報を受信すると（ステップＳ１０５）、全てのプロセッサ・エレメントに対するＲＴペア番号の割り当てと密度行列情報の送信が終了しかたどうか判断し（ステップＳ１０７）、終了していなければ、ステップＳ１０３に戻って、新たなＲＴペア番号の割り当てを決め、そのＲＴペア番号に対応する密度行列情報を、そのプロセッサ・エレメント１４　２に送信する。

【０１２４】同様の操作を繰り返して、ホスト計算機１１は、全てのプロセッサエレメントに対するＲＴペア番号の割り当てと密度行列情報の送信を行い、プロセッサエレメントからのフォック行列の受信を待つ。

【０１２５】ステップＳ１０６で、全てのＲＴペア番号の処理が終了したと判断すると、ホスト計算機１１は、収集したフォック行列情報をもとに、図２６の（数式１４）を解き、新たな係数行列を計算する（ステップＳ１０７）。そして、新たな係数行列とその直前の係数行列とを比較して、自己無撞着な解が得られたかどうか判断する（ステップＳ１０８）。そして、両者が十分良く一致していれば、自己無撞着な解が得られたと判断し、計算を終了する。そうでなければ、ステップＳ１０８からステップＳ１０２に戻り、（数式１８）に従って新たな密度行列を生成し、同様の操作を繰り返す。

【０１２６】なお、通信処理を計算処理でなるべく隠蔽するために、頻繁に用いられるデータのダブル・バッファリングが、ＲＴ並列アルゴリズムにおいても適用可能である。

【０１２７】図５に示すように、各プロセッサ・エレメントが有するメモリのうち、密度行列情報およびフォック行列情報を格納するための領域を、ＡおよびＢのように、２つに分けておく。

【０１２８】そして、密度行列情報格納領域Ａに格納された密度行列要素を用いて計算（以下、この計算をタイプＡの計算と呼ぶ）されたフォック行列要素は、フォック行列情報格納領域Ａに、密度行列情報格納領域Ｂに格納された密度行列要素を用いて計算（以下、この計算をタイプＢの計算と呼ぶ）されたフォック行列要素は、フォック行列情報格納領域Ｂに、それぞれ格納すると定める。

【０１２９】１つのＲＴペア番号に対応したジョブを、タイプＡまたはタイプＢの計算のどちらかに対応させ、タイプＡの計算ジョブの終了後にはタイプＢの計算を、タイプＢの計算ジョブの終了後にはタイプＡの計算を、それぞれ開始する。

【０１３０】タイプＡの計算時には、その直前に行われたタイプＢの計算ジョブによる計算結果がフォック行列情報格納領域Ｂに保持されており、また、密度行列情報格納領域Ｂは自由に上書き可能な領域となっている。

【０１３１】したがって、タイプＡの計算ジョブ実行の最中に、直前の計算結果をフォック行列情報格納領域Ｂから読み出してホスト計算機に送信し、また、次のタイプＢの計算ジョブで用いる情報をホスト計算機から受信して密度行列情報格納領域Ｂに書込むことが可能である。タイプＢの計算ジョブの実行中にも、領域Ａを用いて同様のデータ送受信動作を行なうことができる。

【０１３２】このようなダブル・バッファリング機構を用いることにより、データ送受信処理を計算ジョブの処理中に済ませることができれば、通信処理を計算処理で隠蔽することが可能となり、効率的な処理が行なえる。以下に示す計算例では、いずれもこのダブル・バッファリングを行なっている。

【０１３３】次に、密度行列情報の構成を図６および図７を用いて説明する。

【０１３４】図６に、ホスト計算機１１からプロセッサエレメント１４へ送信する密度行列情報のフォーマットの一例を示す。縮約シェルＲの番号から縮約シェルＷ［Ｎｗ－１］の番号までは整数型データで、縮約シェルＶ［０］に関わる密度行列データブロックから縮約シェルＷ［Ｎｗ－１］に関わる密度行列データブロックは、１つまたは複数の浮動小数点型データ、または、固定小数点型データにより構成されるブロックである。

【０１３５】図６の上から２つの縮約シェルＲおよび縮約シェルＴの番号により、その密度行列情報を受け取ったプロセッサ・エレメント１４が処理すべきＲＴペア番

号が決まる。

【０１３６】縮約シェルＲおよびＴの番号の次は、縮約シェルの集合Ｖおよび縮約シェルの集合Ｗの要素数を表わす２つの整数型データＮｖおよびＮｗである。縮約シェルＶは、それに含まれる縮約基底と縮約シェルＲに含まれる縮約基底の組み合わせのうち、少なくとも１つがａｂでのカットオフで生き残るもの、すなわち縮約シェルＳのループでＳがとりうる縮約シェル番号である。

【０１３７】同様に、縮約シェルＷは、それに含まれる縮約基底と縮約シェルＲに含まれる縮約基底の組み合わせのうち、少なくとも１つがｃｄでのカットオフで生き残るもの、すなわち縮約シェルＵのループでＵがとりうる縮約シェル番号である。

【０１３８】図７に、密度行列データブロックの構成例を示す。これは、縮約シェルＶ［０］に関わる密度行列データブロックを代表例として示したものである。

【０１３９】縮約シェルの軌道量子数が１より大きい場合には、その縮約シェルに関わる密度行列データブロックは、さらに縮約基底に関わるサブブロックにより構成されることになる。図７に示した例は、縮約シェルＶ［０］を構成する縮約基底が、Ｍ０（Ｖ［０］），Ｍ１（Ｖ［０］），・・・，Ｍｍ（Ｖ［０］）というように、ｍ＋１個ある場合であり、それぞれの縮約基底に対応したサブブロックがｍ＋１個ある。

【０１４０】１つのサブブロックは、さらに２つの領域に別れる。１つは、縮約シェルＲを構成する縮約基底Ｉ０（Ｒ）からＩｉ（Ｒ）と、Ｍ０（Ｖ［０］）とをインデックスとする密度行列要素の領域である。もう１つは、縮約シェルＴを構成する縮約基底Ｋ０（Ｔ）からＫｋ（Ｔ）と、Ｍ０（Ｖ［０］）とをインデックスとする密度行列要素の領域である。ここで、ｉ＋１，ｋ＋１は、それぞれ縮約シェルＲ，Ｔを構成する縮約基底の個数であり、これらは、縮約シェルＲ，Ｔの軌道量子数に依存する。これらの領域に含まれる密度行列要素の個数は、縮約シェルＲおよびＴの軌道量子数によって決まり、したがって、１組の密度行列情報の中では、どのサブブロックも同じ大きさとなる。

【０１４１】なお、説明は省略するが、フォック行列情報の構成も同様であり、そのデータ量は密度行列情報と同じである。

【０１４２】［プロセッサ・エレメントへのジョブの割り当て方法の実施の形態の説明］［各ジョブの形成方法および番号付けについて］プロセッサ・エレメントに割り当てられるジョブは、上述の説明から判るように、計算処理のみではなく、ホスト計算機との通信処理をも含んだものとして構成される。

【０１４３】この実施の形態においては、計算処理と通信処理との和からなるジョブの大きさ（サイズ）のばらつきが小さく、できるだけ、ジョブのサイズが均等になるように、各ジョブを形成する。また、各プロセッサ・エレメントが有するメモリの容量には制限があるため、そのメモリに保持する密度行列やフォック行列の要素数がなるべく少なくなるように、各ジョブを形成する。このため、この実施の形態では、次のようにしている。

【０１４４】上述したように、この実施の形態においては、ジョブは、ＲＴペア単位でプロセッサ・エレメントに割り当てられる。そして、ＲＴペアが決まると、縮約シェルＳおよびＵのループを回る回数が決まる。つまり、Ｒが大きい場合には、ＳおよびＵのループを回る回数が多くなり、Ｒが小さい場合には、ＳおよびＵのループを回る回数が少なくなる。そこで、Ｒが大きいジョブには、軌道量子数の高いシェルを割り当て、Ｒが小さいジョブには、軌道量子数の低いシェルを割り当てるようにする。

【０１４５】このようにすれば、ＲＴペアのＲが大きいジョブの場合には、ＳおよびＵのループを回る回数が多くなるが、このジョブに割り当てられる縮約シェルの軌道量子数が低いので、その内部の縮約基底に関するループを回る回数が多くなる。逆に、Ｒが小さいジョブの場合には、ＳおよびＵのループを回る回数が少なくなるが、縮約シェルの軌道量子数が高いのでその内部の縮約基底に関するループを回る回数が多くなる。したがって、ＲＴペアにより定まるジョブに依存した計算量のばらつきは小さくなっている。

【０１４６】また、ＲＴペアのＲの番号が大きいジョブの場合には、図６のＮｖ，Ｎｗが大きくなるが、このジョブに割り当てられる縮約シェルの軌道量子数が低いので、密度行列情報のデータ量が多くなりすぎないようにされているため、その結果としてジョブに依存した通信量のばらつきが小さくなっている。

【０１４７】したがって、ジョブのサイズのばらつきを小さくすることができる。

【０１４８】また、ＲＴペアのＲの番号が大きい場合には、Ｎｖ，Ｎｗが大きくなるため、サブブロックのデータ量を小さくすることで、全体の容量を抑制する必要がある。サブブロックのデータ量は、前述したように、軌道量子数に依存し、軌道量子数の低い縮約シェルに関わるサブブロックはデータ量が少なくなる。

【０１４９】この実施の形態では、ＲＴペアのＲの番号の大きい縮約シェルの軌道量子数は低くされる。逆に、ＲＴペアのＲの番号が小さい場合には、Ｎｖ，Ｎｗが小さいため、サブブロックのデータ量を少なくする必要がないので、番号の小さい縮約シェルの軌道量子数は高くされている。

【０１５０】したがって、このように、軌道量子数の高いシェルが、番号が小さい縮約シェルに、軌道量子数の低いシェルが、番号が大きい縮約シェルに割り当てられることで、容量の小さいメモリであっても、大規模な分子軌道計算への適用が可能となる。

【０１５１】この実施の形態においては、ジョブは、Ｒ

Ｔペア単位でプロセッサ・エレメントに割り当てられるので、ジョブ番号はＲＴペア番号と一致する。ＲＴペア番号は、前述したように、任意に決定することができるが、例えば、図３９の（数式３８）により決定すると、Ｒ、Ｔの大きさの小さいものから順に、ジョブ番号が付与されることになる。

【０１５２】［ジョブ割り当て方法の第１の実施の形態］この例では、図２に示した計算機システムを用いた。そして、この例では、３残基ペプチドであるＧｌｙｃｉｎｅ－Ａｌａｎｉｎｅ－Ｇｌｕｔａｍｉｎｅ（以下、ＧＡＱと記す。分子式：$C_{10}H_{18}N_4O_5$）を計算の対象とし、基底関数として、６－３１１＋＋Ｇ（３ｄ，２ｐ）を用いた。

【０１５３】この例では、縮約シェルの数Ｎｓｈｅｌｌが３３６で、そのうち、軌道量子数が０のｓ軌道のシェルが１６７、軌道量子数が１のｐ軌道のシェルが１１２、軌道量子数が２のｄ軌道のシェルが５７である。

【０１５４】この例では、縮約シェル番号を、０から５６はｄ軌道のシェルに、５７から１６８はｐ軌道のシェルに、１６９から３３５はｓ軌道のシェルに割り当てた。

【０１５５】さらに、この例では、ＲＴペアの番号、すなわち、ジョブ番号は、図３９の（数式３８）により決め、ＲＴペア番号の小さい順にプロセッサ・エレメントへのジョブ割り当てを行なった。

【０１５６】この例により、ＲＴ並列アルゴリズムで、ＳＣＦループを１回だけ実行した場合に要する処理時間とプロセッサ・エレメント数との関係の一例は、図８に示すようなものとなった。

【０１５７】図８において、破線で示した直線は、プロセッサ・エレメント数に反比例して処理時間が減少する理想的な場合を示している。この第１の例では、プロセッサ・エレメント数が増加して１００の付近になると、理想直線から離れ、スケーラブルな並列処理でなくなっていることが判る。

【０１５８】また、図９には、この第１の例において、全体の処理時間のうち、プロセッサ・エレメントが処理を行なっていた時間の占める割合（稼働率）を示したもので、プロセッサ数が１００の付近から稼働率が低下している。

【０１５９】この原因は、プロセッサ・エレメント数の増加に連れて、システム全体の計算処理能力が向上するのに対して、ホスト計算機とプロセッサ・エレメントとの間の通信性能が一定のままで通信量が増大することにより、通信時間が計算時間を越え、プロセッサ・エレメントが、データ受信待ちで計算を行なえない時間が増大するためであると考えられる。

【０１６０】スケーラブルでなくなるもう１つの要因として考えられるのは、並列プロセッサへの計算負荷の割り当てが不均一となっていることである。しかしながら、図１０に示す全プロセッサ・エレメントにおける処理終了時間の分布を見ると、計算負荷の割り当ては、ほぼ均一に行なわれていることがわかる。

【０１６１】並列プロセッサ・エレメントにおける計算時間の合計は、３６３８９秒であり、これを１００プロセッサに均等に割り当てると、３６３．８９秒の計算処理となる。これに対し、通信時間の合計は３６３．０２秒であり、計算時間とほぼ等しくなっている。

【０１６２】しかしながら、図１０に示すように、第１の例の並列処理における各プロセッサ・エレメントにおける計算時間は、４２０秒程度となっている。したがって、第１の例における並列処理は、効率の悪いものであると言える。

【０１６３】図１１に、通信処理と単一のプロセッサ・エレメントにおける計算処理の進行状況を、処理効率の悪い場合と良い場合との比較で模式的に示す。図１１において、網点部が処理を行なっている時間帯である。この網点部の合計の長さは通信と計算とで同じであり、また、効率の悪い場合と良い場合とでも同じである。

【０１６４】効率の悪い処理の場合には、図中、空白部分として示されるように、通信にも計算にも処理を行なっていない時間帯が生じる。そのために、全体としての処理終了までの時間が長くなる。それに対し、効率の良い処理を行なえば、通信処理も計算処理も休みなく行なわれ、処理終了までの時間が短くなる。

【０１６５】上述の第１の例において、ジョブ１つあたりに発生する計算時間と通信時間との関係を図１２に示す。この図１２は、全５６６１６ジョブから、ジョブ番号に関して等間隔に１００個のジョブを抽出してプロットしたものである。図１２に、参照インデックスとして示したように、ジョブにおける縮約シェルＲおよびＴの軌道量子数の組み合わせに応じて、プロットの形状を変えている。例えば、縮約シェルＲの軌道量子数が１すなわちｐ軌道で、縮約シェルＴの軌道量子数が２すなわちｄ軌道の場合の組み合わせ（ｐｄ）は、黒四角の点としてプロットした。

【０１６６】この図１２から判るように、ジョブのサイズのばらつきは小さくても、ジョブによって通信時間と計算時間との割合（比）が異なっている。図１２中の一点鎖線は、通信時間と計算時間の比が０．０１となるラインである。プロセッサ・エレメント数が１００の場合には、通信時間と計算時間との比が、この線より下のジョブが多いと通信量が多くなり、計算処理に待ちが生じることになる。逆に、通信時間と計算時間との比が、この線より上のジョブが多いと計算量が多くなり、通信処理を隠蔽しやすくなる。

【０１６７】前述のように、全体の通信時間と計算時間がほぼ等しい場合には、どの時刻においても、その時点で並列に行われているジョブに関わる通信量と計算量との比の平均値が一点鎖線上に乗るように、ジョブの割り

当てを制御すると、効率のよい並列処理が行える。

【０１６８】なお、この例では、縮約シェル番号の決め方を、前述のように、軌道量子数の高いシェルを小さい番号に、軌道量子数の低いシェルを大きい番号に割り当てるようにして、Ｒが大きい場合に密度行列情報のデータ量が多くなりすぎないようにしているため、その結果としてジョブ番号に依存した通信量のばらつきが小さくなっている。また、前述したように、上述のような縮約シェル番号の決め方をした場合には、ジョブ番号に依存した計算量のばらつきも小さくなっている。

【０１６９】縮約シェル番号を、軌道量子数の高いシェルを小さい番号に、軌道量子数の低いシェルを大きい番号に割り当てるようにしない場合には、計算時間や通信時間のジョブ番号に依存したばらつきが、図１２に示したよりも大きくなり、複数のプロセッサ・エレメントに均等に負荷を割り当てることが、より困難になる。

【０１７０】図１２に示したジョブ当たりの計算時間と通信時間の関係を調べると、一点鎖線よりも上の領域にあるのは、縮約シェルＲ，Ｔともにｓ軌道の場合の全てと、縮約シェルＲがｓ軌道で、縮約シェルＴがｐ軌道の場合の一部であることがわかる。

【０１７１】もう少し詳細に調べると、図３９の（数式３８）により決定したジョブ番号が、約３５０００より大きい場合には、プロットが一点鎖線よりも上にあり、それよりもジョブ番号が小さい場合には、下にあることがわかる。つまり、ＲＴペア番号の小さいジョブは、通信時間の計算時間に対する比が大きく、ＲＴペア番号の大きいジョブは、通信時間の計算時間に対する比が小さい。

【０１７２】以下に説明する第２の実施の形態から第５の実施の形態では、上述のようにして、計算時間や通信時間のばらつきが小さくなるように形成されると共に、図３９の（数式３８）により決定されたジョブ番号のジョブに、上述した図１２に示す性質があることを利用して、より効率の良いジョブの割り当てを行なう。

【０１７３】［ジョブ割り当て方法の第２の実施の形態］上記の考察から、番号の小さいジョブと、番号の大きいジョブとを混合させて割り当てることにより、通信負荷の時間的分散が行なえるものと期待される。そこで、この第２の実施の形態では、図１３に示すような、割り当て待ち行列を作成して、並列プロセッサ・エレメント２あるいは１４への割り当てを行う。

【０１７４】図１３に示した例においては、説明を簡単にするために、全ジョブ数Ｎが３３の場合の割り当て待ち行列を示したものである。図１３において、括弧の外の数字がジョブ番号で、これが分子軌道計算におけるＲＴ並列アルゴリズムではＲＴペア番号に相当する。また、括弧内の数字は、ジョブ割り当てを行なう順序を示している。

【０１７５】ホスト計算機は、この待ち行列を参照して、ジョブ番号を、この待ち行列の上から順に読み出しながら、プロセッサ・エレメントへのジョブの割り当てを行なう。

【０１７６】ここで、最初のジョブは、番号が０のジョブとし、それ以降は、前の番号をＪとすると、

・ＪがＮ／２以下なら、番号がＮ－Ｊ－１のジョブ

・ＪがＮ／２以上なら、番号がＮ－Ｊのジョブ

を、プロセッサ・エレメントに次に割り当てるジョブとする。

【０１７７】このようにすることにより、ＲＴペア番号の小さい、通信時間と計算時間との比の大きいジョブと、ＲＴペア番号の大きい、通信時間と計算時間との比が小さいジョブとが交互に割り当てられるようになる。したがって、複数のプロセッサ・エレメントで並列に行なわれているジョブに関わる通信量と計算量との比の平均が、プロセッサ・エレメントの数分の１、プロセッサ・エレメントの数が１００の場合には０．０１、に近くなり、効率的な並列処理が行なえる。

【０１７８】図１４に、第２の実施の形態のジョブの割り当て方法による並列処理方法の計算における処理時間とプロセッサ・エレメント数との関係を示す。この例の計算の対象とした分子、用いた基底関数は、ともに図８に結果を示した、第１の実施の形態におけるものと同じで、それぞれ、ＧＡＱと、６－３１１＋＋Ｇ（３ｄ，２ｐ）である。

【０１７９】図８の結果と比較して、プロセッサ・エレメント数が１００付近における計算時間が、約９％短くなり、前述の第１の実施の形態の場合より高速な処理が可能となった。

【０１８０】図１５は、この第２の実施の形態の場合におけるプロセッサ・エレメントの稼働率とプロセッサ・エレメント数との関係を示したもので、第１の実施の形態の説明における図９に対応するものである。

【０１８１】両者の比較から判るように、第２の実施の形態の場合には、プロセッサ・エレメント数が１００の付近において、第１の実施の形態の例と比較して稼働率が高くなっている。

【０１８２】また、この第２の実施の形態において、プロセッサ・エレメント数が１００の場合における、各プロセッサ・エレメントの処理終了時間を図１６に示す。これは、第１の実施の形態の図１０に対応するものである。並列プロセッサ・エレメントに均等に負荷が分配され、終了時間のばらつきが小さくなっていることが判る。

【０１８３】なお、割り当て待ち行列の先頭のジョブを、ＲＴペア番号が最大のジョブとし、それ以降は、前の番号をＪとした場合には、

・ＪがＮ／２以下なら、番号がＮ－Ｊのジョブ

・ＪがＮ／２以上なら、番号がＮ－Ｊ－１のジョブ

を、プロセッサ・エレメントに次に割り当てるジョブと

する。この場合にも、上述と同様の効果が得られる。

【０１８４】また、任意の番号のジョブを先頭とし、それ以降は、前の番号をＪとすると、

・ＪがＮ／２以下なら、番号がＮ－Ｊ－１のジョブ

・ＪがＮ／２以上なら、番号がＮ－Ｊのジョブ

を次に割り当てるジョブとし、その番号が０未満または最大のＲＴペア番号よりも大きくなったら最大のＲＴペア番号の半分の番号のジョブから再スタートし、同様の手続きで次のジョブ番号を決める、などの類似の手法を用いて割り当て待ち行列を作成することによっても、同様の効果が得られる。

【０１８５】なお、第２の実施の形態において、次に割り当てるジョブの番号を決める際の条件に記述に用いている除算は、商を小数点以下まで求める正確な除算であり、第３の実施の形態以降で用いている整数の除算とは異なるものである。

【０１８６】［ジョブ割り当て方法の第３の実施の形態］第３の実施の形態では、複数個のプロセッサ・エレメントをグループ分けすると共に、それに対応して、複数個のジョブも同数のグループに分け、各グループに対して、それぞれ割り当て待ち行列を用意する。

【０１８７】すなわち、複数個のプロセッサ・エレメントは、グループ内のプロセッサ・エレメントの数がほぼ等しいＧ個（Ｇは２以上の整数）のプロセッサ・グループに分割すると共に、複数個のジョブを、グループ内のジョブの数がほぼ等しい、プロセッサ・グループの数と同数のＧ個のジョブ・グループに分割する。

【０１８８】そして、プロセッサ・グループと、ジョブ・グループとを１対１に対応させて、一つのジョブ・グループ内のジョブを、対応する一つのプロセッサ・グループ内のプロセッサ・エレメントに割り当てるようにする。

【０１８９】その割り当て方法としては、まず、通信時間と演算時間とからなるジョブサイズと、通信時間と演算時間との比とが近似するジョブ同士は、ジョブ・グループの異なるものにそれぞれ属するようにする割り当てる。さらに、ジョブサイズと、通信時間と演算時間との比とが近似するジョブは、他のプロセッサ・グループでの割り当て順序と重ならないように、複数のジョブ・グループのそれぞれ内での割り当て順番を、それぞれのジョブ・グループにおいて互いに異なるように割り当てる。

【０１９０】グループ数Ｇが４、ジョブ数Ｎが３３の簡単な場合について、図１７および図１８を用いて、この第３の実施の形態の場合の複数個のジョブ・グループに１対１に対応する複数個の待ち行列の作成方法の具体例を説明する。

【０１９１】前述例と同様に、これらの図１７、図１８において、括弧外の番号はジョブ番号すなわちＲＴペア番号を示し、括弧内の番号は同じ待ち行列に割り当てられたジョブ・グループ内での割り当てを行なう順序を示す。

【０１９２】まず、図１７に示すように、ジョブ番号Ｊを、ジョブ・グループ数（待ち行列数）Ｇで除算した時の余りに相当する番号ｇのグループに、各ジョブを分配する。ここで、各待ち行列内で、一旦、この分配されたジョブを番号順に並べる。

【０１９３】次に、ジョブ番号ＪをグループＧで除算（小数点以下を切り捨てる整数の除算を使用、以下同様）した時の商によりブロック分けを行う。図１７の例では、各ブロック番号ｂは、その商である。

【０１９４】ここで、各ブロック番号ｂに属する複数のジョブは、ジョブ番号が近接したものであり、図１２に示したように、互いにジョブサイズが近似し、かつ、計算時間と計算時間との比が近似したものとなり、それらのジョブが、それぞれ異なる待ち行例に分配されることになる。

【０１９５】全ジョブ数Ｎをグループ数Ｇで除算した時の商をＢとした時、全ジョブ数Ｎがグループ数Ｇの倍数の場合には、ブロック数はＢに、そうでない場合には、Ｂ＋１となる。後者の場合には、最後のブロックにジョブ番号のある待ち行列と、最後のブロックにジョブ番号がない待ち行列とが存在する。

【０１９６】次に、ｇ番目のジョブ・グループ（ｇ番目の待ち行列）内のジョブのうち、Ｂを、グループ数Ｇで除算した商をｘとして、ｘ×ｇ以上、ｘ×（ｇ＋１）未満の番号のジョブ・ブロックに含まれるジョブを、最初のジョブとする。これは、同じブロック内のジョブが、複数個のジョブ・グループで同じ順番とならないようにするためである。

【０１９７】図１４の例では、Ｂ＝Ｎ／Ｇ＝３３／４＝８であり、ｘ＝Ｂ／Ｇ＝８／４＝２であるから、待ち行列０番のジョブ・グループであれば、ｘ×ｇ＝２×０＝０以上、ｘ×（ｇ＋１）＝２×１＝２未満の番号０または１のブロックから、開始ジョブを選択することになる。この例では、上記条件に適合する最小の番号を、開始ジョブとして選択することとした。

【０１９８】以下に示す計算例でもそのようにしたが、条件に適合する番号から任意に開始番号を選択しても良い。

【０１９９】次に、各ジョブ・グループにおいて、直前に割り当てが済んだジョブがｂ番目のジョブ・ブロックに含まれる場合には、ｂ＋１番目のジョブ・ブロック内のジョブを次のジョブ番号とする。但し、ｂ＋１番目のジョブ・ブロック内にｇ番目のグループ内のジョブがなければ、０番目のジョブ・ブロック内のジョブを次のジョブとする。

【０２００】図１４の例における、待ち行列１番のジョブ・グループでは、ｂ＝６番目のブロックに含まれる番号２５のジョブの次のジョブは、ｂ＋１＝７番目のブロ

ックに含まれる番号２９のジョブ、その次のジョブは（８番目のブロックに含まれる１番目のグループのジョブがないので）０番目のブロック内の番号１のジョブとなる。

【０２０１】なお、この例では、次のジョブをｂ＋１番目のジョブ・ブロック内のジョブとしたが、次のジョブをｂ－１番目のジョブ・ブロック内のジョブとしてもよい。この場合、ｂ－１が負となる時には、ｇ番目のグループのジョブが含まれる最大の番号のジョブ・ブロック内のジョブを次のジョブとする。

【０２０２】以上のように順序を決め、各ジョブ・グループ内の割り当て順序に従ってジョブ番号を並べ直した、各ジョブ・グループの待ち行列を図１８に示す。

【０２０３】図１９に、第３の実施の形態の計算における処理時間とプロセッサ・エレメント数との関係を示す。

【０２０４】計算の対象とした分子、用いた基底関数は、第１の実施の形態で用いたものと同じである。なお、各プロセッサ・エレメントの演算処理能力は互いに等しいものとし、５個のプロセッサ・エレメントで１つのプロセッサ・グループを構成した。

【０２０５】この図１９から判るように、この第３の実施の形態によれば、図８の結果と比較して、プロセッサ・エレメント数が１００付近における計算時間が約１２％短くなり、第２の実施の形態よりさらに高速な処理が可能となる。

【０２０６】図２０は、この第３の実施の形態の場合の計算例におけるプロセッサ・エレメントの稼働率とプロセッサ・エレメント数との関係を示したもので、第１の実施の形態の説明における図９に対応するものである。これによれば、プロセッサ・エレメント数が１００の付近において、第１の実施の形態と比較して稼働率が高くなり、１００％に近くなっている。

【０２０７】また、この第３の実施の形態において、プロセッサ・エレメント数が１００の場合における、各プロセッサ・エレメントの処理終了時間を図２１に示す。終了時間のばらつきが４秒程度あるものの、並列プロセッサ・エレメントにほぼ均等に負荷が分配されていることが判る。

【０２０８】［ジョブ割り当て方法の第４の実施の形態］第４の実施の形態は、プロセッサ・エレメントをグループ分けし、各グループに対して割り当て待ち行列を用意する点では、前述の第３の実施の形態と同様であるが、プロセッサ・エレメントのグループ分けの方法が、第３の実施の形態と異なる。

【０２０９】グループ数Ｇが４、ジョブ数Ｎが３３の簡単な場合について、図２２および図２３を用いて、この第４の実施の形態の場合の複数個のジョブ・グループのそれぞれに対応する待ち行列の作成方法を説明する。なお、図２２、図２３において、括弧の外の番号はジョブ番号すなわちＲＴベア番号を示し、括弧内の番号は同じ待ち行列に割り当てられたジョブ・グループ内で割り当てを行なう順序を示すのは、前述の実施の形態の場合と同様である。

【０２１０】まず、図２２に示すように、ジョブ番号Ｊをジョブ・グループ数（待ち行列数）Ｇの２倍の２×Ｇで除算した時の余りがｇまたは２×Ｇ－ｇ－１に等しいジョブが、ｇ番目のグループに含まれるように、各ジョブを分配する。そして、分配したジョブを、各待ち行列内で、一旦、番号順に並べる。

【０２１１】次に、ジョブ番号Ｊをグループ数Ｇで除算（小数点以下を切り捨てる整数の除算を使用、以下同様）した時の商によりブロック分けを行う。図２２において、各ブロック番号ｂは、その商である。

【０２１２】全ジョブ数Ｎをグループ数Ｇで除算した時の商をＢとした時、全ジョブ数Ｎがグループ数Ｇの倍数の場合には、ブロック数はＢに、そうでない場合にはＢ＋１となる。後者の場合、最後のブロックにジョブ番号のある待ち行列と、最後のブロックにジョブ番号がない待ち行列とが存在する。

【０２１３】次に、ｇ番目のジョブ・グループ（ｇ番目の待ち行列）内のジョブのうち、Ｂをグループ数Ｇで除算した商をｘとして、ｘ×ｇ以上、ｘ×（ｇ＋１）未満の番号のジョブ・ブロックに含まれるジョブを、最初のジョブとする。これは、同じブロック内のジョブが、複数個のジョブ・グループで同じ順番とならないようにするためである。

【０２１４】図２２の例では、Ｂ＝Ｎ／Ｇ＝３３／４＝８であり、ｘ＝Ｂ／Ｇ＝８／４＝２であるから、待ち行列０番目のグループであれば、ｘ×ｇ＝２×０＝０以上、ｘ×（ｇ＋１）＝２×１＝２未満の番号０または１のブロックから、開始ジョブを選択することになる。この例では、上記条件に適合する最小の番号を、開始ジョブとして選択することとした。

【０２１５】以下に示す計算例でもそのようにしたが、条件に適合する番号から任意に開始番号を選択しても良い。

【０２１６】次に、各ジョブ・グループにおいて、直前に割り当てが済んだジョブがｂ番目のジョブ・ブロックに含まれる場合には、ｂ＋１番目のジョブ・ブロック内のジョブを次のジョブ番号とする。但し、ｂ＋１番目のジョブ・ブロック内にｇ番目のグループ内のジョブがなければ、０番目のジョブ・ブロック内のジョブを次のジョブとする。

【０２１７】図２２の例における待ち行列１番目のグループでは、ｂ＝６番目のブロックに含まれる番号２５のジョブの次のジョブは、ｂ＋１＝７番目のブロックに含まれる番号３０のジョブ、その次のジョブは（８番目のブロックに含まれる１番目のグループのジョブがないので）０番目のブロック内の番号１のジョブとなる。

【０２１８】なお、この例では、次のジョブを、ｂ＋１番目のジョブ・ブロック内のジョブとしたが、次のジョブを、ｂ－１番目のジョブ・ブロック内のジョブとしてもよい。この場合、ｂ－１が負となる時には、ｇ番目のグループのジョブが含まれる最大の番号のジョブ・ブロック内のジョブを次のジョブとする。

【０２１９】以上のように順序を決め、各ジョブ・グループ内の割り当て順序に従ってジョブ番号を並べ直した、各ジョブ・グループの待ち行列を図２３に示す。

【０２２０】図２４に、第４の実施の形態の計算における処理時間とプロセッサ・エレメント数との関係を示す。計算の対象とした分子、用いた基底関数は、第１の実施の形態で用いたものと同じである。なお、各プロセッサ・エレメントの演算処理能力は互いに等しいものとし、５個のプロセッサ・エレメントで１つのプロセッサ・グループを構成した。

【０２２１】図２４から判るように、この第４の実施の形態によれば、図８の結果と比較して、プロセッサ・エレメント数が１００付近における計算時間が約１２％短くなり、第２の実施の形態よりさらに高速な処理が可能となる。

【０２２２】図２５は、この第４の実施の形態の場合の計算例におけるプロセッサ・エレメントの稼働率とプロセッサ・エレメント数との関係を示したもので、第１の実施の形態の説明における図９に対応するものである。これによれば、プロセッサ・エレメント数が１００の付近において、第１の実施の形態と比較して稼働率が高くなり１００％に近くなっている。

【０２２３】また、この第４の実施の形態において、プロセッサ・エレメント数が１００の場合における、各プロセッサ・エレメントの処理終了時間を図２６に示す。終了時間のばらつきが３．５秒程度あるものの、並列プロセッサ・エレメントにほぼ均等に負荷が分配されている。

【０２２４】［ジョブ割り当て方法の第５の実施の形態］第５の実施の形態は、プロセッサ・エレメントおよびジョブを、１対１にグループ分けし、各グループに対して割り当て待ち行列を用意する点では、第３および第４の実施の形態と同様であり、また、プロセッサ・エレメントのグループ分けの方法が第３の実施の形態と同様であるが、グループ内の割り当て順序の決め方が第３および第４の実施の形態と異なる。

【０２２５】グループ数Ｇが４、ジョブ数Ｎが６５、グループ当たりのプロセッサ・エレメント数が２の簡単な場合について、図２７および図２８を用いて、この第５の実施の形態の場合の複数個のジョブ・グループのそれぞれに対応する待ち行列の作成方法を説明する。なお、これらの図２７、図２８において、括弧の外の番号はジョブ番号すなわちＲＴペア番号を示し、括弧内の番号は同じ待ち行列に割り当てられたジョブ・グループ内で割り当てを行なう順序を示すのは、前述の実施の形態の場合と同様である。

【０２２６】まず、図２７に示すように、ジョブ番号Ｊをジョブ・グループ数（待ち行列数）Ｇで除算した時の余りがｇに等しいジョブが、ｇ番目のグループに含まれるように、各ジョブを分配する。この分配方法は、第３の実施の形態と同様であるが、第４の実施の形態と同様の方法で分配してもよい。ここで、分配したジョブを、各待ち行列内では、一旦、番号順に並べる。

【０２２７】次に、ジョブ番号Ｊをグループ数Ｇで除算（小数点以下を切り捨てる整数の除算を使用、以下同様）した時の商によりブロック分けを行う。図２２において、各ブロック番号ｂは、その商である。

【０２２８】全ジョブ数Ｎをグループ数Ｇで除算した時の商をＢとした時、全ジョブ数Ｎがグループ数Ｇの倍数の場合には、ブロック数はＢに、そうでない場合にはＢ＋１となる。後者の場合、最後のブロックにジョブ番号のある待ち行列と、最後のブロックにジョブ番号がない待ち行列とが存在する。

【０２２９】次に、ｇ番目のジョブ・グループ（ｇ番目の待ち行列）内のジョブのうち、ＢをＧで除算した商をｘとし、さらにｘをグループに割り当てられるプロセッサ・エレメントの数ｍで除算した商をｙとして、ｙ×ｇ以上、ｙ×（ｇ＋１）未満の番号のジョブ・ブロックに含まれるジョブを、最初のジョブとする。これは、同じブロック内のジョブが、複数個のジョブ・グループで同じ順番とならないようにするためである。

【０２３０】図２７の例では、Ｂ＝Ｎ／Ｇ＝６５／４＝１６であり、ｘ＝Ｂ／Ｇ＝１６／４＝４、ｙ＝ｘ／ｍ＝４／２＝２であるから、待ち行列０番目のグループであれば、ｙ×ｇ＝２×０＝０以上、ｙ×（ｇ＋１）＝２×１＝２未満の番号０または１のブロックから、開始ジョブを選択することになる。この例では、上記条件に適合する番号のうち最も小さい番号を、開始ジョブとして選択した。

【０２３１】以下に示す計算例でも上記条件に適合する最小の番号を選択するようにしたが、条件に適合する番号から任意に開始番号を選択しても良い。また、開始ジョブを選択するブロックの条件を、第３の実施の形態あるいは第４の実施の形態と同様にしてもよい。

【０２３２】次に、各ジョブ・グループにおいて、直前に割り当てが済んだジョブがｂ番目のジョブ・ブロックに含まれる場合には、ｂ＋Ｂ／ｍ番目のジョブ・ブロック内のジョブを次のジョブ番号とする。但し、ｂ＋Ｂ／ｍ番目のジョブ・ブロック内にｇ番目のグループ内のジョブがなければ、ｂ＋１を（Ｂ／ｍ）で除算した余りに等しい番号のジョブ・ブロック内のジョブを次のジョブとする。

【０２３３】図２７の例における待ち行列１番目のグループでは、ｂ＝７番目のブロックに含まれる番号２９の

ジョブの次のジョブは、ｂ＋（Ｂ／ｍ）＝７＋（１６／２）＝１５番目のブロックに含まれる番号６１のジョブとなる。その次のジョブは、ｂ＝１５として、ｂ＋（Ｂ／ｍ）＝１５＋（１６／２）＝２３番目のブロックに含まれる１番目のグループのジョブがないので、ｂ＋１＝１６を、（Ｂ／ｍ）＝８で除算した余りに等しい０番目のブロック内の番号１のジョブとなる。

【０２３４】なお、この例では、次のジョブを、ｂ＋Ｂ／ｍ番目のジョブ・ブロック内のジョブとしたが、次のジョブを、ｂ－Ｂ／ｍ番目のジョブ・ブロック内のジョブとしてもよい。この場合、ｂ－Ｂ／ｍが負で、かつ、ｂが０であれば、（Ｂ／ｍ）で除算した余りが（Ｂ／ｍ）－１に等しくなる番号のジョブ・ブロックのうち、ｇ番目のグループのジョブを含む最大の番号のジョブ・ブロック内のジョブを次のジョブとし、ｂ－Ｂ／ｍが負で、かつ、ｂが０でなければ、（Ｂ／ｍ）で除算した余りがｂ－１を（Ｂ／ｍ）で除算した余りと等しくなるようなジョブ・ブロック番号のうちｇ番目のジョブ・グループのジョブを含む最大の番号のジョブ・ブロック内のジョブを次のジョブとする。

【０２３５】このように順序を決め、グループ内の割り当て順序に従ってジョブ番号を並べ直したのが、図２８に示す待ち行列である。

【０２３６】図２９に、第５の実施の形態の計算における処理時間とプロセッサ・エレメント数との関係を示す。計算の対象とした分子、用いた基底関数は、第１の実施の形態で用いたものと同じである。なお、各プロセッサ・エレメントの演算処理能力は互いに等しいものとし、５個のプロセッサ・エレメントで１つのプロセッサ・グループを構成した。

【０２３７】図２９から判るように、この第５の実施の形態によれば、図８の結果と比較して、プロセッサ・エレメント数が１００付近における計算時間が約１３％短くなり、第３の実施の形態および第４の実施の形態よりさらに高速な処理が可能となった。

【０２３８】図３０は、この計算例におけるプロセッサ・エレメントの稼働率とプロセッサ・エレメント数との関係を示したもので、第１の実施の形態の説明における図９に対応するものである。これによれば、プロセッサ・エレメント数が１００の付近において、従来例と比較して稼働率が高くなりほぼ１００％となっている。

【０２３９】また、この第５の実施の形態において、プロセッサ・エレメント数が１００の場合における、各プロセッサ・エレメントの処理終了時間を図３１に示す。終了時間のばらつきが１．５秒程度と小さく、並列プロセッサ・エレメントに均等に負荷が分配されていることが確認できた。

【０２４０】以上に説明した第３、第４、第５の実施の形態においては、同一の演算処理性能を有する５個のプロセッサ・エレメントによりプロセッサ・グループを構成したが、プロセッサ・エレメントの個数はこれに限る必要はない。

【０２４１】また、全てのプロセッサ・エレメントの演算処理性能が同じである必要もなく、例えば１つのプロセッサ・グループを２倍の演算処理性能を有するプロセッサ・エレメント２つで構成し、他のプロセッサ・グループを１倍の演算処理性能を有するプロセッサ・エレメント４つで構成するなど、プロセッサ・グループ内の合計の演算処理性能がグループ間で同一となっていれば良い。

【０２４２】また、上述の実施の形態では、非経験的分子軌道計算においてＲＴ並列アルゴリズムを用いる場合を例として、この発明の説明を行なったが、他の並列処理においても、この発明を適用することが可能である。

【０２４３】

【発明の効果】以上説明したように、この発明による並列処理方法における各プロセッサへのジョブ割り当て方法を用いると、効率の良い並列処理が可能となり、システム全体の処理時間を短くすることができる。

【図面の簡単な説明】

【図１】この発明の実施の形態で用いる計算機システムの一例を示すブロック図である。

【図２】この発明の実施の形態で用いる計算機システムの他の例を示すブロック図である。

【図３】この発明の実施の形態のＲＴ並列アルゴリズムをプログラム・コード形式で示す図である。

【図４】この発明の実施の形態のＲＴ並列アルゴリズムのフローチャートを示す図である。

【図５】この発明の実施の形態で用いるデータのダブル・バッファリングを行なう場合のメモリ領域の使い方を説明するための図である。

【図６】この発明の実施の形態において、ホスト計算機からプロセッサエレメントへ送信される密度行列情報のフォーマットの一例を示す図である。

【図７】この発明の実施の形態における密度行列データブロックの構成例を示す図である。

【図８】この発明の第１の実施の形態の分子軌道計算における並列プロセッサ数とＳＣＦ計算の処理時間の関係を示す図である。

【図９】この発明の第１の実施の形態の分子軌道計算における並列プロセッサ数とプロセッサの稼働率の関係を示す図である。

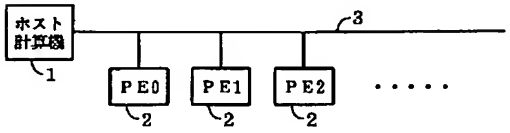
【図１０】この発明の第１の実施の形態の分子軌道計算における個々の並列プロセッサの処理の終了時間を示す図である。

【図１１】効率の悪い処理方法と効率の良い処理方法における通信処理と計算処理の進行状況を示す模式図である。

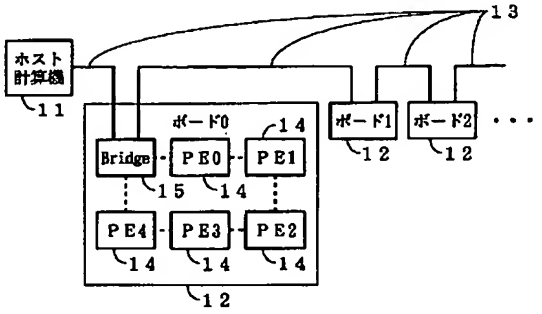
【図１２】この発明の第１の実施の形態の分子軌道計算における、ジョブあたりに発生する通信量と計算量の相

関関係を示す図である。
【図１３】この発明の第２の実施の形態で用いる、プロセッサ・エレメントへのジョブの割り当てのための待ち行列の例を示す図である。
【図１４】この発明の第２の実施の形態における、並列プロセッサ数とＳＣＦ計算の処理時間の関係を示す図である。
【図１５】この発明の第２の実施の形態における、並列プロセッサ数とプロセッサの稼働率の関係を示す図である。
【図１６】この発明の第２の実施の形態における、個々の並列プロセッサの処理の終了時間を示す図である。
【図１７】この発明の第３の実施の形態におけるジョブ・グループ分割方法およびジョブの割り当て方法を説明するための図である。
【図１８】この発明の第３の実施の形態における、プロセッサ・エレメントへのジョブの割り当てのための待ち行列の例を示す図である。
【図１９】この発明の第３の実施の形態における、並列プロセッサ数とＳＣＦ計算の処理時間の関係を示す図である。
【図２０】この発明の第３の実施の形態における、並列プロセッサ数とプロセッサの稼働率の関係を示す図である。
【図２１】この発明の第３の実施の形態における、個々の並列プロセッサの処理の終了時間を示す図である。
【図２２】この発明の第４の実施の形態におけるジョブ・グループ分割方法およびジョブの割り当て方法を説明するための図である。
【図２３】この発明の第４の実施の形態における、プロセッサ・エレメントへのジョブの割り当てのための待ち行列の例を示す図である。
【図２４】この発明の第４の実施の形態における、並列プロセッサ数とＳＣＦ計算の処理時間の関係を示す図である。
【図２５】この発明の第４の実施の形態における、並列プロセッサ数とプロセッサの稼働率の関係を示す図である。
【図２６】この発明の第４の実施の形態における、個々の並列プロセッサの処理の終了時間を示す図である。
【図２７】この発明の第５の実施の形態におけるジョブ・グループ分割方法およびジョブの割り当て方法を説明するための図である。
【図２８】この発明の第５の実施の形態における、プロセッサ・エレメントへのジョブの割り当てのための待ち行列の例を示す図である。
【図２９】この発明の第５の実施の形態における、並列プロセッサ数とＳＣＦ計算の処理時間の関係を示す図である。
【図３０】この発明の第５の実施の形態における、並列プロセッサ数とプロセッサの稼働率の関係を示す図である。
【図３１】この発明の第５の実施の形態における、個々の並列プロセッサの処理の終了時間を示す図である。
【図３２】原始基底関数と、その角運動量、軌道指数、原子核座標との対応例を示す図である。
【図３３】非経験的分子軌道計算法の説明に用いる数式を示す図である。
【図３４】非経験的分子軌道計算法の説明に用いる数式を示す図である。
【図３５】非経験的分子軌道計算法の説明に用いる数式を示す図である。
【図３６】非経験的分子軌道計算法の説明に用いる数式を示す図である。
【図３７】非経験的分子軌道計算法の説明に用いる数式を示す図である。
【図３８】非経験的分子軌道計算法の説明に用いる数式を示す図である。
【図３９】非経験的分子軌道計算法の説明に用いる数式を示す図である。
【符号の説明】
１、１１　ホスト計算機
２、１４　プロセッサ・エレメント
３、１３　バス

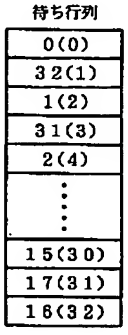
【図１】



【図２】



【図１３】

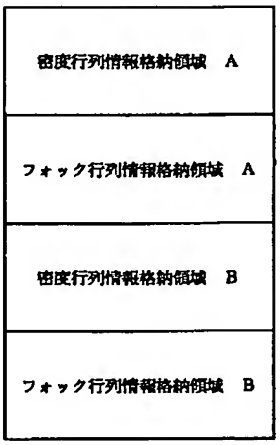


【図３】

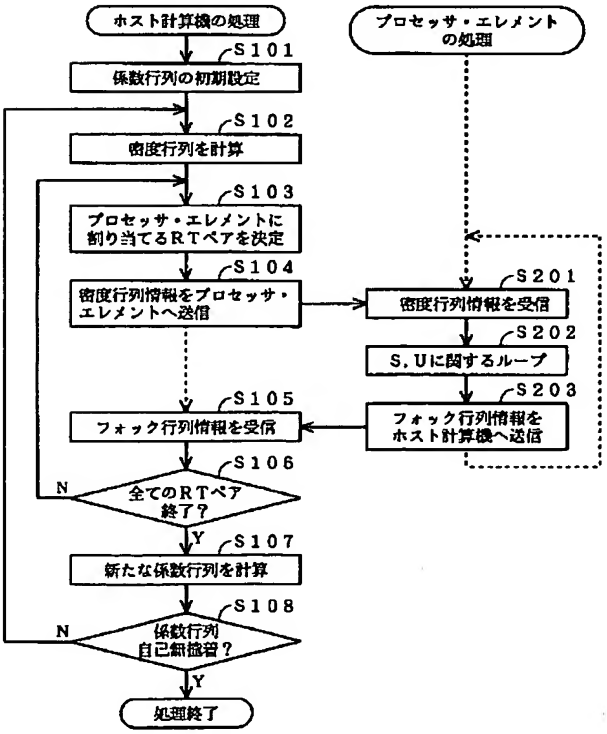
RT並列アルゴリズム（この発明の実施形態）

```
for(R=0;R<Nshell;R++){
for(T=0;T<=R;T++){
                  for(S=0;S<=R;S++){
                  for(U=0;U<=R;U++){
                    for(I=b_basis(R);I<=e_basis(R);I++){
                    for(J=b_basis(S);J<=e_basis(S);J++){
                    for(K=b_basis(T);K<=e_basis(T);K++){
                    for(L=b_basis(U);L<=e_basis(U);L++){
                      G_IJKL=G(I,J,K,L);
                      F[I][J]+=P[K][L]*G_IJKL;
                      if(L<=K&&K<I){
                        if(J<I)  F[K][L]+=2*P[I][J]*G_IJKL;
                        else     F[K][L]+=P[I][J]*G_IJKL;
                      }
                      F[I][L]-=0.5*P[K][J]*G_IJKL;
                      if(J<I&&K<=J)
                               F[K][J]-=0.5*P[I][L]*G_IJKL;
                      if(K<I&&J<=K&&L<I)
                               F[K][J]-=0.5*P[I][L]*G_IJKL;
                    }}}}
                  }}
}}
```

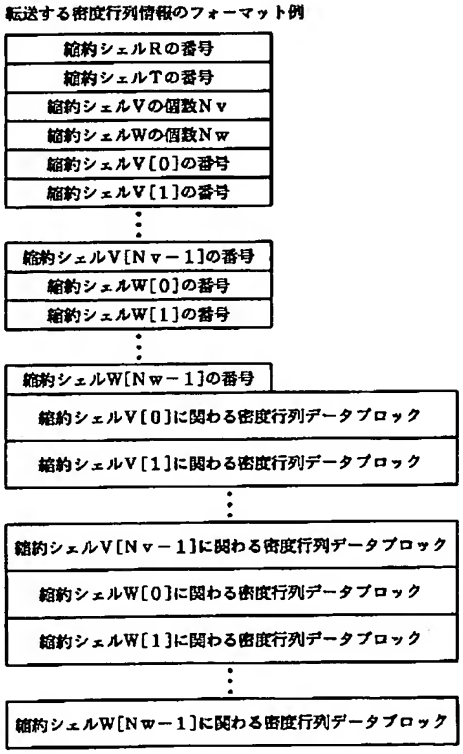
【図５】



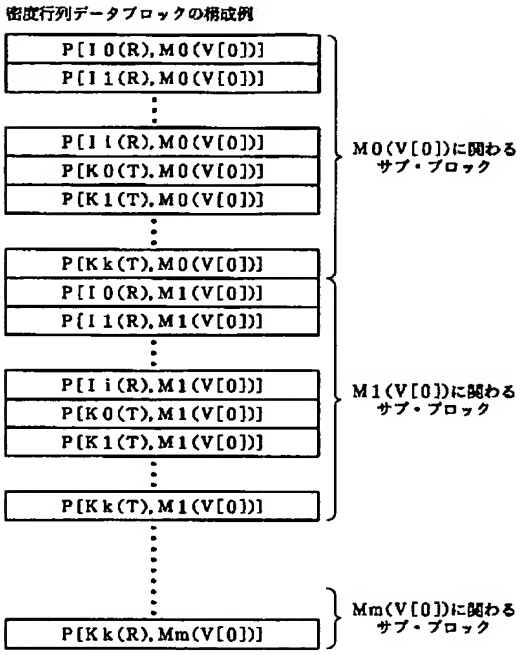
【図４】



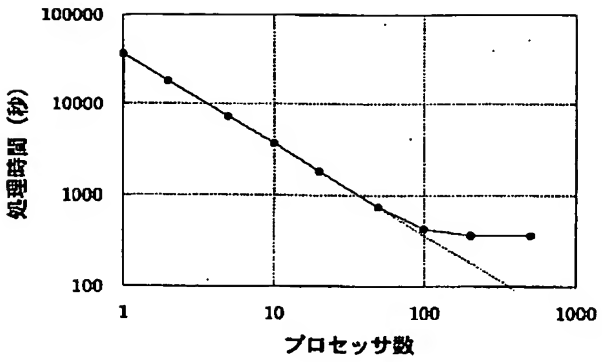
【図６】



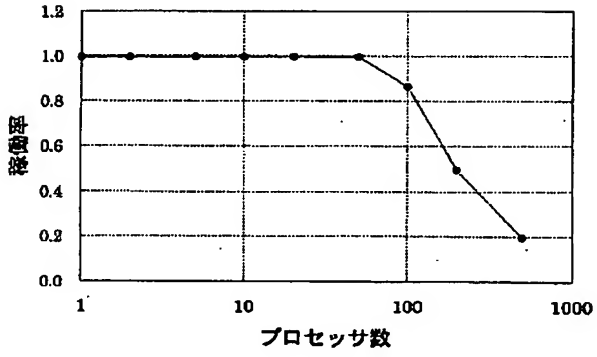
【図７】



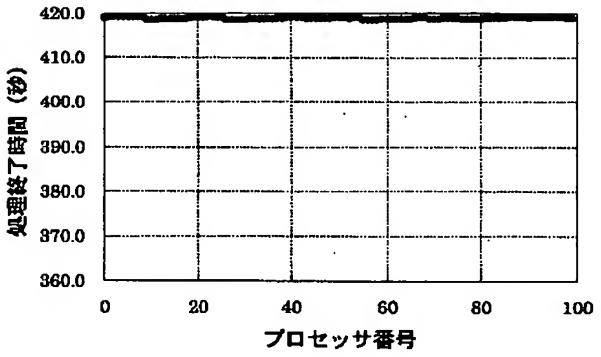
【図８】



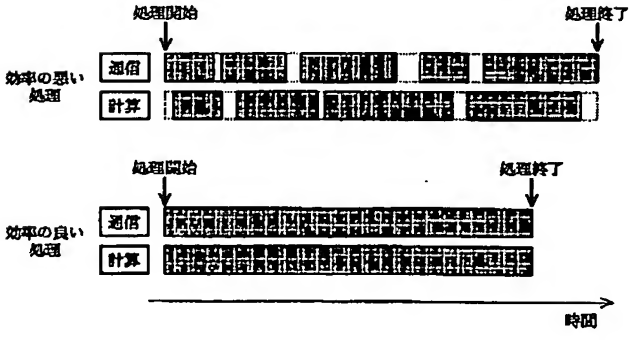
【図９】



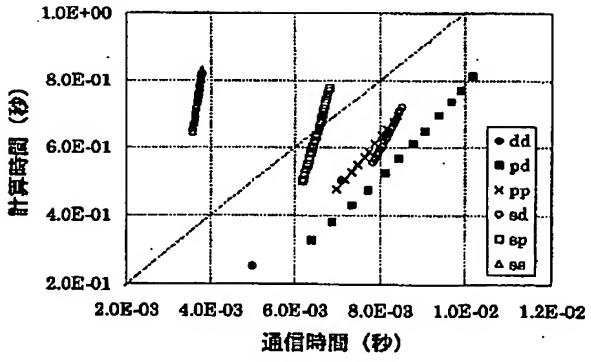
【図１０】



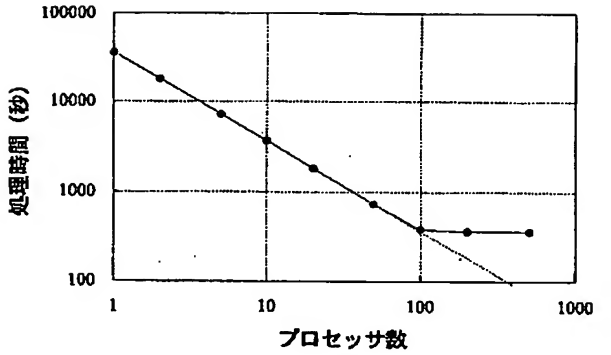
【図１１】



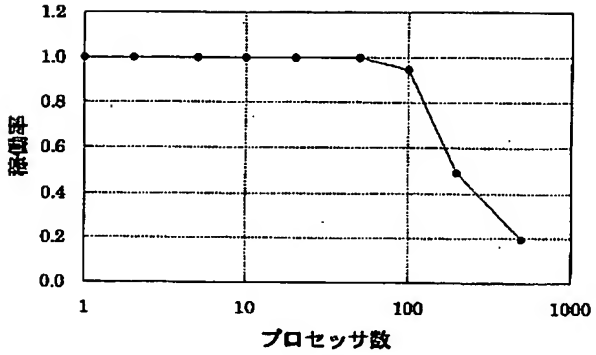
【図１２】



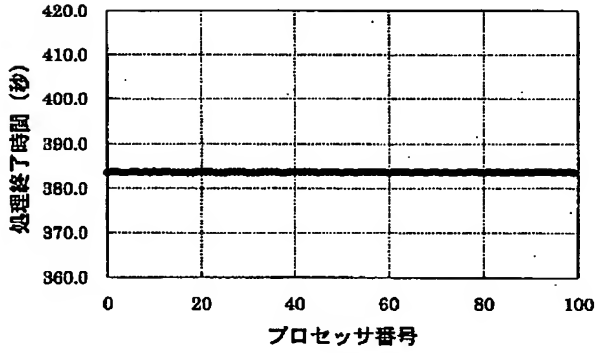
【図１４】



【図１５】



【図１６】



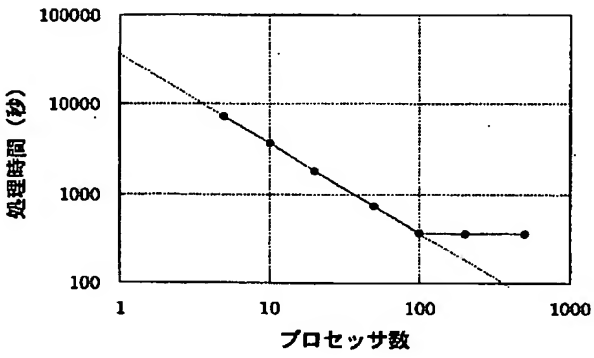
【図１７】

	待ち行列0番のジョブ	待ち行列1番のジョブ	待ち行列2番のジョブ	待ち行列3番のジョブ
ブロック0	0(0)	1(6)	2(4)	3(2)
ブロック1	4(1)	5(7)	6(5)	7(3)
ブロック2	8(2)	9(0)	10(6)	11(4)
ブロック3	12(3)	13(1)	14(7)	15(5)
ブロック4	16(4)	17(2)	18(0)	19(6)
ブロック5	20(5)	21(3)	22(1)	23(7)
ブロック6	24(6)	25(4)	26(2)	27(0)
ブロック7	28(7)	29(5)	30(3)	31(1)
ブロック8	32(8)			

【図１８】

待ち行列0番	待ち行列1番	待ち行列2番	待ち行列3番
0(0)	9(0)	18(0)	27(0)
4(1)	13(1)	22(1)	31(1)
8(2)	17(2)	26(2)	3(2)
12(3)	21(3)	30(3)	7(3)
16(4)	25(4)	2(4)	11(4)
20(5)	29(5)	6(5)	15(5)
24(6)	1(6)	10(6)	19(6)
28(7)	5(7)	14(7)	23(7)
32(8)			

【図１９】

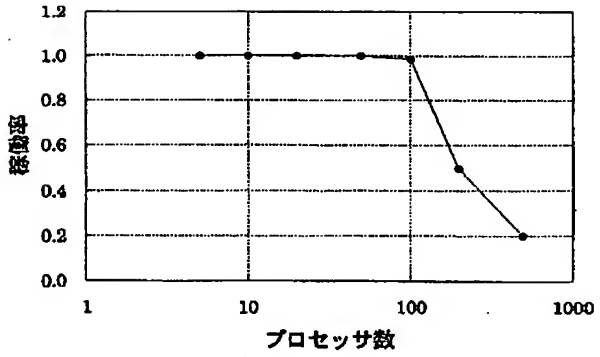


【図３２】

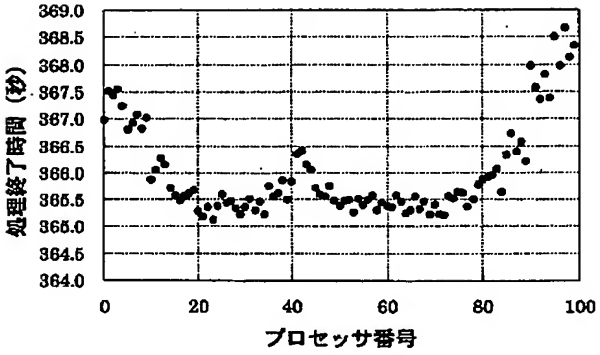
【表１　原始基底関数ｉ，ｊ，ｋ，ｌの角運動量、軌道指数、原子核座標】

原始基底関数の番号	角運動量	軌道指数	原子核座標
i	a	ζ_a	A
j	b	ζ_b	B
k	c	ζ_c	C
l	d	ζ_d	D

【図２０】



【図２１】



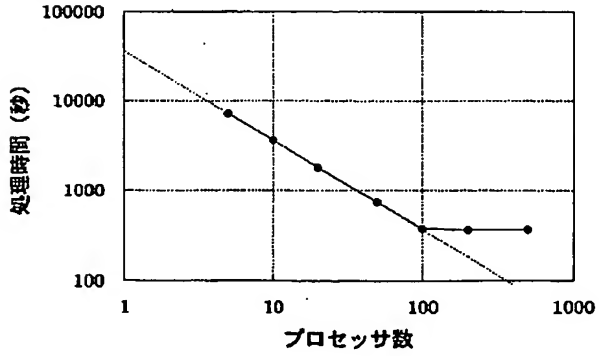
【図２２】

	待ち行列 0番のジョブ	待ち行列 1番のジョブ	待ち行列 2番のジョブ	待ち行列 3番のジョブ
ブロック0	0(0)	1(6)	2(4)	3(2)
ブロック1	7(1)	6(7)	5(5)	4(3)
ブロック2	8(2)	9(0)	10(6)	11(4)
ブロック3	15(3)	14(1)	13(7)	12(5)
ブロック4	16(4)	17(2)	18(0)	19(6)
ブロック5	23(5)	22(3)	21(1)	20(7)
ブロック6	24(6)	25(4)	26(2)	27(0)
ブロック7	31(7)	30(5)	29(3)	28(1)
ブロック8	32(8)			

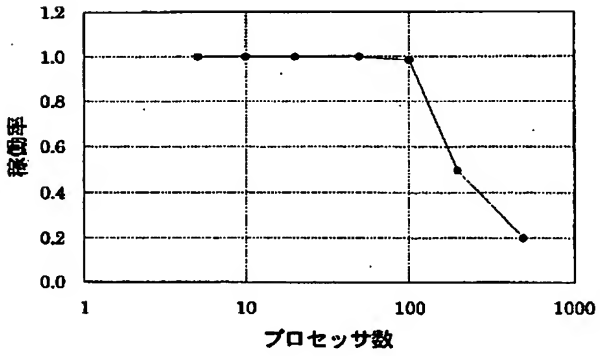
【図２３】

待ち行列 0番	待ち行列 1番	待ち行列 2番	待ち行列 3番
0(0)	9(0)	18(0)	27(0)
7(1)	14(1)	21(1)	28(1)
8(2)	17(2)	26(2)	3(2)
15(3)	22(3)	29(3)	4(3)
16(4)	25(4)	2(4)	11(4)
23(5)	30(5)	5(5)	12(5)
24(6)	1(6)	10(6)	19(6)
31(7)	6(7)	13(7)	20(7)
32(8)			

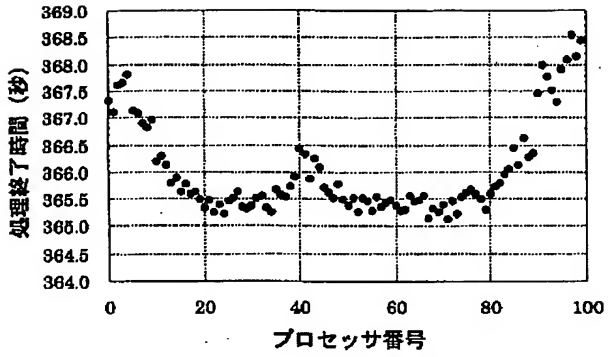
【図２４】



【図２５】



【図２６】



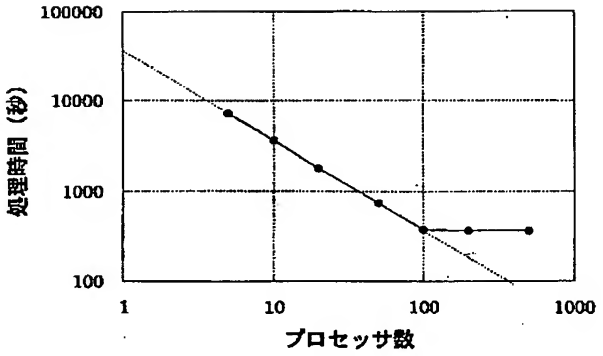
【図２７】

	待ち行列0番のジョブ	待ち行列1番のジョブ	待ち行列2番のジョブ	待ち行列3番のジョブ
ブロック0	0(0)	1(12)	2(8)	3(4)
ブロック1	4(3)	5(14)	6(10)	7(6)
ブロック2	8(5)	9(0)	10(12)	11(8)
ブロック3	12(7)	13(2)	14(14)	15(10)
ブロック4	16(9)	17(4)	18(0)	19(12)
ブロック5	20(11)	21(6)	22(2)	23(14)
ブロック6	24(13)	25(8)	26(4)	27(0)
ブロック7	28(15)	29(10)	30(6)	31(2)
ブロック8	32(1)	33(13)	34(9)	35(5)
ブロック9	36(4)	37(15)	38(11)	39(7)
ブロック10	40(6)	41(1)	42(13)	43(9)
ブロック11	44(8)	45(3)	46(15)	47(11)
ブロック12	48(10)	49(5)	50(1)	51(13)
ブロック13	52(12)	53(7)	54(3)	55(15)
ブロック14	56(14)	57(9)	58(5)	59(1)
ブロック15	60(16)	61(11)	62(7)	63(3)
ブロック16	64(2)			

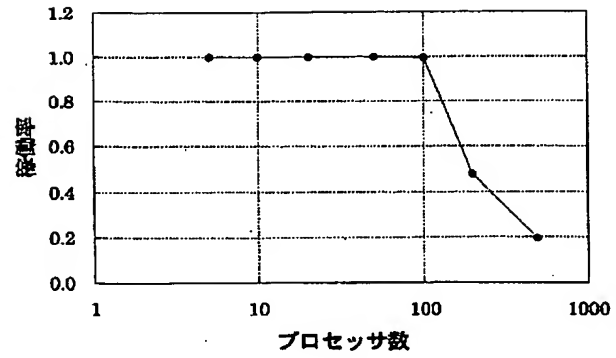
【図２８】

待ち行列0番	待ち行列1番	待ち行列2番	待ち行列3番
0(0)	9(0)	18(0)	27(0)
32(1)	41(1)	50(1)	59(1)
64(2)	13(2)	22(2)	31(2)
4(3)	45(3)	54(3)	63(3)
36(4)	17(4)	26(4)	3(4)
8(5)	49(5)	58(5)	35(5)
40(6)	21(6)	30(6)	7(6)
12(7)	53(7)	62(7)	39(7)
44(8)	25(8)	2(8)	11(8)
16(9)	57(9)	34(9)	43(9)
48(10)	29(10)	6(10)	15(10)
20(11)	61(11)	38(11)	47(11)
52(12)	1(12)	10(12)	19(12)
24(13)	33(13)	42(13)	51(13)
56(14)	5(14)	14(14)	23(14)
28(15)	37(15)	46(15)	55(15)
60(16)			

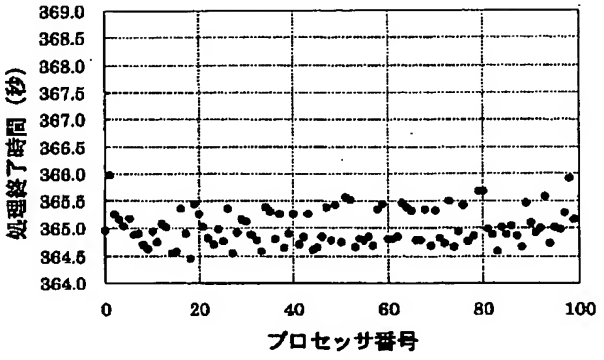
【図２９】



【図３０】



【図３１】



【図３３】

(数式1)

$$\phi_\mu = \sum_I \chi_I C_{I\mu}$$

(数式2)

$$\Psi(1,2,\cdots,2n) = \frac{1}{\sqrt{(2n)!}} \times \begin{vmatrix} \phi_1\alpha(1) & \phi_1\beta(1) & \phi_2\alpha(1) & \phi_2\beta(1) & \cdots & \phi_n\alpha(1) & \phi_n\beta(1) \\ \phi_1\alpha(2) & \phi_1\beta(2) & \phi_2\alpha(2) & \phi_2\beta(2) & \cdots & \phi_n\alpha(2) & \phi_n\beta(2) \\ \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots \\ \phi_1\alpha(2n) & \phi_1\beta(2n) & \phi_2\alpha(2n) & \phi_2\beta(2n) & \cdots & \phi_n\alpha(2n) & \phi_n\beta(2n) \end{vmatrix}$$

(数式3)

$$H = H_1 + H_2$$

(数式4)

$$H_1 = \sum_p \left(-\frac{1}{2}\nabla_p^2 - \sum_A \frac{Z_A}{r_{pA}} \right)$$

(数式5)

$$H_2 = \sum_p \sum_{q(>p)} \frac{1}{r_{pq}}$$

【図３４】

(数式6)

$$\varepsilon = \int \Psi H \Psi \, d\tau = \int \Psi (H_1 + H_2) \Psi \, d\tau$$

$$= 2\sum_\mu H_\mu + \sum_\mu \sum_\nu (2J_{\mu\nu} - K_{\mu\nu})$$

(数式7)

$$H_\mu = \int \phi_\mu(1) h^{core}(1) \phi_\mu(1) \, d\tau_1$$

(数式8)

$$J_{\mu\nu} = \int\int \phi_\mu(1) \phi_\mu(1) \frac{1}{r_{12}} \phi_\nu(2) \phi_\nu(2) \, d\tau_1 \, d\tau_2$$

(数式9)

$$K_{\mu\nu} = \int\int \phi_\mu(1) \phi_\nu(1) \frac{1}{r_{12}} \phi_\mu(2) \phi_\nu(2) \, d\tau_1 \, d\tau_2$$

【図３５】

(数式１０)

$$\varepsilon = 2\sum_{\mu}\left(\sum_{I}\sum_{J} C_{I\mu}C_{J\mu}H_{IJ}\right)$$

$$+\sum_{\mu}\sum_{\nu}\left\{\sum_{I}\sum_{J}\sum_{K}\sum_{L} C_{I\mu}C_{J\mu}C_{K\nu}C_{L\nu}\left[2G(I,J,K,L)-G(I,K,J,L)\right]\right\}$$

(数式１１)

$$H_{IJ}=\int \chi_I(1)\, h^{core}(1)\, \chi_J(1)\, d\tau_1$$

(数式１２)

$$h^{core}(1)=-\frac{1}{2}\nabla_1^2-\sum_{A}\frac{Z_A}{r_{1A}}$$

(数式１３)

$$G(I,J,K,L)=\int\int \chi_I(1)\,\chi_J(1)\frac{1}{r_{12}}\chi_K(2)\,\chi_L(2)\,d\tau_1 d\tau_2$$

【図３６】

(数式１４)

$$\sum_{J}(F_{IJ}-\varepsilon_{\mu}S_{IJ})C_{J\mu}=0$$

(数式１５)

$$F_{IJ}=H_{IJ}+\sum_{K}\sum_{L}P_{KL}\left[G(I,J,K,L)-\frac{1}{2}G(I,K,J,L)\right]$$

(数式１６)

$$\varepsilon_{\mu}=H_{\mu}+\sum_{\nu}(2J_{\mu\nu}-K_{\mu\nu})$$

(数式１７)

$$S_{IJ}=\int \chi_I(1)\,\chi_J(1)\,d\tau_1$$

(数式１８)

$$P_{KL}=2\sum_{\nu}C_{K\nu}C_{L\nu}$$

(数式１９)

$$G(I,J,K,L)=G(I,J,L,K)=G(J,I,K,L)=G(J,I,L,K)=$$
$$G(K,L,I,J)=G(L,K,I,J)=G(K,L,J,I)=G(L,K,J,I)$$

【図３７】

(数式２０)

$$\chi(r, n, R) = (r_x - R_x)^{n_x}(r_y - R_y)^{n_y}(r_z - R_z)^{n_z} \times \sum_m d_m \exp\left[-\zeta_m (r-R)^2\right]$$

(数式２１)

$$\phi(r, n, R) = (r_x - R_x)^{n_x}(r_y - R_y)^{n_y}(r_z - R_z)^{n_z} \exp\left[-\zeta (r-R)^2\right]$$

(数式２２)

$$\sum_m d_m \exp\left[-\zeta_m (r-R)^2\right]$$

(数式２３)

$$G(I, J, K, L) = \sum_{m1}\sum_{m2}\sum_{m3}\sum_{m4} d_{m1} d_{m2} d_{m3} d_{m4} g(i, j, k, l)$$

(数式２４)

$$g(i, j, k, l) = \int\int \phi_i(1)\phi_j(1)\frac{1}{r_{12}}\phi_k(2)\phi_l(2)\, d\tau_1 d\tau_2$$

(数式２５)

$$g(i, j, k, l) = g(i, j, l, k) = g(j, i, k, l) = g(j, i, l, k) = g(k, l, i, j) = g(l, k, i, j) = g(k, l, j, i) = g(l, k, j, i)$$

【図３８】

(数式２６)

$$[0_a 0_b, 0_c 0_d]^{(m)} = \frac{1}{\sqrt{\zeta+\eta}} \cdot K(\zeta_a, \zeta_b, A, B) \cdot K(\zeta_c, \zeta_d, C, D) \cdot F_m(T)$$

(数式２７)

$$F_m(T) = \int t^{2m} \exp\left(-T t^2\right) dt$$

(数式２８)

$$T = \rho (P-Q)^2$$

(数式２９)

$$K(\zeta, \zeta', R, R') = \sqrt{2}\frac{\pi^{5/4}}{\zeta+\zeta'} \exp\left[-\frac{\zeta\zeta'(R-R')^2}{\zeta+\zeta'}\right]$$

(数式３０)

$$\zeta = \zeta_a + \zeta_b$$

(数式３１)

$$\eta = \zeta_c + \zeta_d$$

(数式３２)

$$\rho = \frac{\zeta\eta}{\zeta+\eta}$$

(数式３３)

$$P = \frac{\zeta_a A + \zeta_b B}{\zeta_a + \zeta_b}$$

(数式３４)

$$Q = \frac{\zeta_c C + \zeta_d D}{\zeta_c + \zeta_d}$$

【図３９】

(数式３５)

$$[(a+1_i)\,b,\,c\,d]^{(m)} = (P_i - A_i)\,[a\,b,\,c\,d]^{(m)} + (W_i - P_i)\,[a\,b,\,c\,d]^{(m+1)} + \frac{N_i(a)}{2\zeta}\left\{[(a-1_i)\,b,\,c\,d]^{(m)} - \frac{\rho}{\zeta}[(a-1_i)\,b,\,c\,d]^{(m+1)}\right\} + \frac{N_i(b)}{2\zeta}\left\{[a\,(b-1_i),\,c\,d]^{(m)} - \frac{\rho}{\zeta}[a\,(b-1_i),\,c\,d]^{(m+1)}\right\} + \frac{N_i(c)}{(\zeta+\eta)}[a\,b,\,(c-1_i)\,d]^{(m+1)} + \frac{N_i(d)}{(\zeta+\eta)}[a\,b,\,c\,(d-1_i)]^{(m+1)}$$

(数式３６)

$$W = \frac{\zeta P + \eta Q}{\zeta + \eta}$$

(数式３７)

$$\exp\left[-\frac{\zeta\zeta'(R-R')^2}{\zeta+\zeta'}\right]$$

(数式３８)

$$\frac{R(R+1)}{2} + T$$

フロントページの続き

(72)発明者　稲畑　深二郎
神奈川県足柄上郡中井町境430　グリーンテクなかい　富士ゼロックス株式会社内
(72)発明者　宮川　宣明
神奈川県足柄上郡中井町境430　グリーンテクなかい　富士ゼロックス株式会社内
(72)発明者　高島　一
東京都豊島区高田３－24－１　大正製薬株式会社内
(72)発明者　北村　一泰
東京都豊島区高田３－24－１　大正製薬株式会社内
(72)発明者　長嶋　雲兵
東京都東久留米市金山町２－10－４
Ｆターム(参考)　5B045 GG02 GG15
5B098 AA10 GA03 GC08 GC16 GD02 GD14